
# WAIS Server
# and
# WAIS Workstation
# for UNIX

## Administrator's Manual
## Release 1.0

A Wide Area Information Server System

WAIS Inc.'s licensor(s) make no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the software. WAIS Inc.'s licensor(s) does not warrant, guarantee or make any representations regarding the use or the results of the use of the software in terms of its correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by you. The exclusion of implied warranties is not permitted by some states. The above exclusion may not apply to you.
In no event will WAIS Inc's licensor(s), and their directors, officers, employees or agents (collectively WAIS Inc.'s licensor) be liable to you for any consequential, incidental or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use the software even if WAIS Inc.'s licensor has been advised of the possibility of such damages, because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. WAIS Inc.'s licensor's liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to $50.



Printed in the United States of America.

# Table of Contents

# How to Use This Manual

The *Administrator's Manual* describes how to use the WAIS™ Server and WAIS Workstation products for UNIX®. It includes step-by-step procedures for how to set up your WAIS network publishing system.

## Who Should Read this Manual

The *Administrator's Manual* is intended primarily for administrators of large collections of data that wish to make this data available online over a local or wide-area computer network. Some familiarity with the UNIX operating system is desirable.

## Organization

The *Administrator's Manual* will help you build WAIS databases and help you to set up and maintain a WAIS server. The major sections are:

- *Chapter 1: Introduction* - an overview of the WAIS software suite. — no pages!
- *Chapter 2: Installation* - how to set up the WAIS software.
- *Chapter 3: The WAIS Directory Structure* - A guided tour of how the databases and the server are organized.
- *Chapter 4: Building a WAIS Database* - an introduction to setting up WAIS databases.
- *Chapter 5: Setting Up a WAIS Server* - how to run a server for your databases.
- *Chapter 6: Customizing a WAIS Database* - how to use additional features of the WAIS database to meet the unique needs of your database.
- *Chapter 7: Monitoring WAIS Usage* - how to keep track of your server.
- *Chapter 8: Reference Section* - a short reference section on each of the WAIS programs.

- *Appendix A: WAIS Quick Start* - a crash course in WAIS software installation.
- *Appendix B: Recommended Reading* - a list of helpful references for UNIX administration.
- *Appendix C: Default Stopword List* - a list of the default stopwords used by the software.
- *Appendix D: Glossary of WAIS Terms* - a list of the most frequently used WAIS terms and their definitions.

This manual will lead you through the steps of setting up and maintaining a WAIS server, and will introduce some of the considerations required in managing a server. If you are an experienced UNIX user, you may want to skip ahead to Appendix A: WAIS Quick Start. This section is a crash course on getting a server up and running. Once it is set up you can return here to get a better understanding of the WAIS network publishing system.

The format of the *Administrator's Manual* is an extended tutorial. If you follow it from start to finish you'll have a working WAIS server, and a good understanding of the WAIS software. Later you can come back to it and use it as a reference manual.

# Conventions Used in this Manual

In this manual, command names and their literal arguments are printed in boldface fixed-width font, and non-literal arguments are printed in italics. For example:

```
waisreporter log-file-name -d /wais/data/mydata
```

Non-literal arguments, such as *log-file-name*, should be filled in with the value most appropriate for you. Lines printed by the computer are in fixed-width font:

```
parsing /wais/data/jargon.231.text
```

Actual commands typed at a UNIX prompt are preceded with the prompt, '%' character, where the prompt character is not typed. An example usage of the **waisreporter** command might be displayed as:

```
% waisreporter /wais/logs/server.log
```

If an actual command requires more than a single line to display, multiple lines may be used to clearly illustrate the usage. In this case, it is assumed that the line is treated as a single command, with no intermediate carriage return characters. For example.

```
% waisparse -parse text /wais/data/resumes/Smith.txt
           -parse tiff /wais/data/resumes/Smith.tif
```

is meant to be interpreted as a single command line.

# Software Release Naming Conventions

WAIS software releases are named using the following conventions:

*product-majorversion-minorversion-release-platform*

Thus, ws-1-0-10-next is release 10 of the WAIS Workstation version 1.0 for NeXT™ computers. Throughout this manual, we will use a more generic name, wais-1-0-10, to represent the WAIS Server and WAIS Workstation products on all platforms.

# 1

# Introduction

## What is WAIS?

WAIS is a network publishing system for helping users find answers to questions from information sources over a computer network. The question may be expressed in natural language or boolean syntax, and the information sources may be local or remote. WAIS is the mechanism where, by asking a question, a user can search and retrieve documents from information sources all over the world.

## WAIS Architecture

The WAIS network publishing architecture has four main components: the client, the protocol, the server and the database. The client is a user-interface program that requests services from remote or local servers. The server is a program that services these requests. The server generally runs on a machine physically connected to disks containing one or more information source, or database. The protocol is used to connect WAIS clients and servers and is based on the NISO Z39.50 standard.

The goal of the WAIS network publishing system is to create an open architecture of information servers and clients by using a standard computer-to-computer protocol that enables users to find and question servers.

## The WAIS Software

The process of creating a WAIS database, and serving it to remote clients is illustrated in Figure 1. Following the black arrows from left to right, the flow of information can be traced from its original

repository, to its destination in response to a client's search. Taken together, the original data and the WAIS index make up a complete WAIS database. The **waislookup** command is used for testing the database. The server executes the **waisserver** (or **waisworkstation**) program which references the WAIS database, and returns answers to the user's question. The server also writes logs that are summarized by the **waisreporter** program.

Now consider the alternative path, from the client, asking a question of the server. The dotted arrows illustrate how a user's question and relevant documents are fed to the server. The server examines the index, which refers it to the original data. This produces a list of headlines which the server returns to the client. The same path is followed when the client makes a retrieval request. This time however, actual records from the original data are returned.

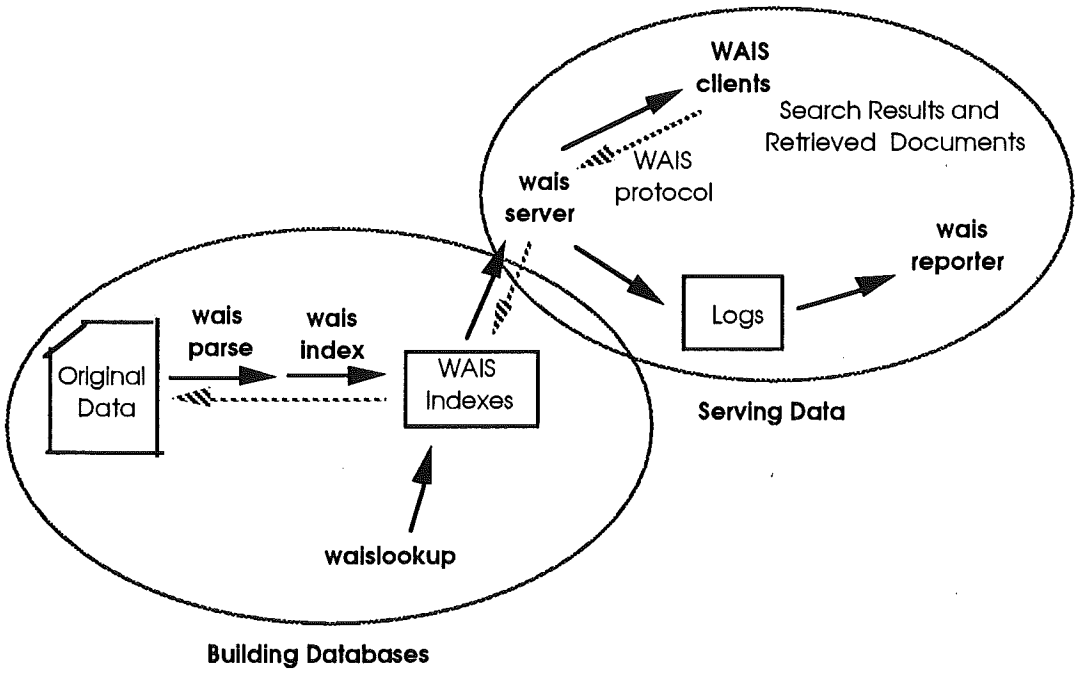


*Figure 1: Components of the WAIS System*

The important relationships are as follows:

- The **waisparse** and **waisindex** programs build a WAIS database from one or more data files which you provide.
- WAIS clients and servers communicate using the WAIS protocol.

- The **waisserver** program examines the WAIS index and the original data when it responds to a user's question.
- The WAIS database is based on the original data, and must be updated whenever that data is modified or moved.

The rest of this manual will describe how these parts fit together in the WAIS system. You will find sections on how to build and customize your WAIS database, install a server, and monitor the usage of your server.

# Getting Help

If you have problems or questions regarding the installation or use of the WAIS software, please contact us at:

WAIS support through electronic mail: support@wais.com
WAIS support hotline for WAIS Server customers: 415-327-WAIS
WAIS Fax: 415-327-6513
Address: WAIS Inc, 1040 Noel Drive, Menlo Park, CA 94025

# Electronic Services

There are a several electronic mailing lists available to users of the WAIS software. With these mailing lists, you can receive assistance from the WAIS Inc technical support staff, and the from the experienced users of the WAIS software.

**support@wais.com**: Prompt online technical support for the WAIS Inc product line. WAIS Inc actively encourages administrators and users alike to use this list to seek help on any issue related to the WAIS Inc products. This mailing list welcomes suggestions, comments, questions, and bug reports.

**wais-discussion@think.com**: Weekly digest of mail from users and developers on network publishing (includes all wais-interest postings). Requests for inclusion to this mailing list should be made to wais-discussion-request@think.com. Anonymous **ftp** access to the mailing-list archives can be found in the following public directory: /pub/wais/wais-discussion/issue*@quake.think.com

**wais-talk@think.com**: An interactive list for WAIS software developers wishing to share their insights and questions with other developers. This is a fairly active mailing list, with an average of several messages a day. Requests for inclusion to this mailing list should be made to wais-talk-request@think.com.

**comp.infosystems.wais**: A network news discussion group on WAIS issues. All postings to wais-discussion@think.com go to this group as well.

# 2

# Installation

This chapter describes the contents of a WAIS software release, and shows you how to set up WAIS software on your machine.

## Getting Started

Installing WAIS software is fairly easy, but there are a few considerations you must keep in mind before you get started. This chapter identifies them, and you can use it as a worksheet as you do your installation. Don't worry if you don't know the answer to some of the questions. You can revisit this section once you've read more of the manual.

Before you start, make sure that you have the following:

- A UNIX machine running SunOS 4.1.1, 4.1.2, 4.1.3, 5.1, or NeXTstep 3.0, 3.1.
- Disk space of approximately 5 megabytes in which to load the WAIS software.
- Basic knowledge of UNIX. Appendix B includes references on UNIX administration that you may find useful.
- An account on a UNIX server machine.
- Superuser access to the UNIX server machine. This is only required for running the server in daemon mode. For more information, see Chapter 5, Setting up a WAIS Server.
- A TCP/IP network to connect WAIS clients.

During the installation, you'll need to make the following decisions. The questions will be fully explained in following sections. They are listed here to help plan your installation.

- Where should the software be installed?

- What data would work best in your WAIS database?
- Which machine should you use as your WAIS server?
- What TCP/IP port should the server run on?
- Who should have access to the databases?

# Checking the Contents

Before you begin the software installation process, make sure you have a full WAIS release. It should contain the following documentation:

- *WAIS Server and WAIS Workstation for UNIX Administrator's Manual* (this manual), and
- *WAIS for Macintosh User's Manual*

The software package itself is released in one of three forms:

- If you've **ftp**'d the software from WAIS Inc's **ftp** site, it will be in a compressed archive file on your hard disk. The name will be something like wais-1-0-10.tar.Z.
- If you've received a tape from us, the software will be in an archive on the tape.
- You may also have a Macintosh or DOS disk containing licensed WAIS client software and documentation.

If any of these components are missing, please contact us at your earliest convenience.

# Unpacking your WAIS System

The first step is getting the WAIS software onto your UNIX machine. At this point you'll need to decide which machine should become the WAIS server, and which directory (and disk) should hold the software. There are several considerations in deciding which machine to use as the server.

- The machine's network location should match your distribution plans. If you intend to publish your WAIS servers to the Internet community, the machine should be directly on the Internet. If you are setting your server up for use as a

confidential resource, you may want to put it on a network that is protected from the Internet by a firewall machine.

- The machine's network location and name should be fairly stable, since hundreds or thousands of clients may come to depend on the server running on the machine.
- We highly recommend that the WAIS databases be located on disks mounted directly (locally) on the server machine. The WAIS server can use databases on disks remotely mounted through NFS, but performance is greatly reduced.
- The data which is indexed by WAIS should also be locally mounted, although this is less of a concern than the location of the WAIS index files.
- The class of machine and amount of memory should be appropriate for the server task it is to perform. Requirements are highly variable depending on the number of users and the size and character of the database. As a general rule, server machines should have at least 16 Megabytes of RAM. If the server is used lightly, other processes can easily share the machine.

Once you've decided on a machine to run the WAIS server, you'll need to decide where to put the software release. Our recommended directory layout is described in detail in the next chapter. The directory you're building now will be the root of the WAIS software tree. Here are the criterion for making your decision.

- The release itself requires about 5 megabytes. You may want to leave some spare room on the release disk for future WAIS releases and upgrades. You don't need to have room for all your databases on a single disk.
- The name of the release directory should be fairly stable, as clients will come to depend on it.

In our examples we'll use /wais as the WAIS directory. Whenever you see /wais, substitute the name you've chosen.

Once you have made your decision, create the directory, and move into it. If it already exists you won't need to do the **mkdir**'s again.

```
% mkdir /wais
% cd /wais
```

If you've **ftp**'d the software over the Internet, you'll need to uncompress it and unpack it. The file is compressed if its filename ends in .Z, and it is an archive if it has a .tar extension. The UNIX **zcat** program uncompresses the file, and passes the results on to **tar**, the UNIX Tape ARchive utility. Here's the line that does the work:

```
% mv wais-1-0-10.tar.Z /wais
% cd /wais
% zcat wais-1-0-10.tar.Z | tar -xvf -
```

If the software is on a tape, you'll have to copy the tape onto your hard disk. That means you will need to know which UNIX device describes the tape drive on your system. Your system administrator should be able to help with this. We will use a typical one in this example; it is the first tape drive on the system, called /dev/rst0 on SunOS 4.1.x, and /dev/rdsk/c0t4d0s2 on SunOS 5.1. The WAIS software tape is not compressed, so all you need to do is **tar**.

```
% cd /wais
% tar -xvf /dev/rst0
```

In either case, the computer should print a list of the contents of the WAIS release. It will look something like this, although it will vary depending on the exact release you have received.

```
x wais-1-0-10/bin/waisindex, 344064 bytes, 672 tape blocks
x wais-1-0-10/bin/waisparse, 172032 bytes, 336 tape blocks
x wais-1-0-10/bin/waisserver, 745472 bytes, 1456 tape blocks
x wais-1-0-10/bin/waisreporter, 114688 bytes, 224 tape blocks
x wais-1-0-10/bin/waislookup, 458752 bytes, 896 tape blocks
x wais-1-0-10/bin/README, 106 bytes, 1 tape blocks
x wais-1-0-10/bin/waissearch, 507904 bytes, 992 tape blocks
x wais-1-0-10/util/rmnl, 73728 bytes, 144 tape blocks
x wais-1-0-10/util/byte-split, 24576 bytes, 48 tape blocks
x wais-1-0-10/util/ebcdic2ascii, 24576 bytes, 48 tape blocks
x wais-1-0-10/util/tmpl, 73728 bytes, 144 tape blocks
x wais-1-0-10/util/update-lcl-dir-of-servers, 1024 bytes, 2 tape blocks
x wais-1-0-10/util/run-waisreporter, 1226 bytes, 3 tape blocks
x wais-1-0-10/util/stopwords.txt, 2002 bytes, 4 tape blocks
x wais-1-0-10/util/README, 206 bytes, 1 tape blocks
x wais-1-0-10/doc/byte-split.txt, 689 bytes, 2 tape blocks
x wais-1-0-10/doc/rmnl.txt, 675 bytes, 2 tape blocks
x wais-1-0-10/doc/access-lists.txt, 1726 bytes, 4 tape blocks
x wais-1-0-10/doc/ethics.txt, 9338 bytes, 19 tape blocks
x wais-1-0-10/doc/waisworkstation.txt, 4817 bytes, 10 tape blocks
x wais-1-0-10/doc/waisindex.txt, 5659 bytes, 12 tape blocks
x wais-1-0-10/doc/waisparse.txt, 9185 bytes, 18 tape blocks
x wais-1-0-10/doc/waisserver.txt, 4817 bytes, 10 tape blocks
x wais-1-0-10/doc/ebcdic2ascii.txt, 588 bytes, 2 tape blocks
x wais-1-0-10/doc/tmpl.txt, 1814 bytes, 4 tape blocks
x wais-1-0-10/doc/waislookup.txt, 3416 bytes, 7 tape blocks
x wais-1-0-10/doc/waisreporter.txt, 1987 bytes, 4 tape blocks
x wais-1-0-10/doc/waissearch.txt, 3414 bytes, 7 tape blocks
x wais-1-0-10/man/byte-split.1, 531 bytes, 2 tape blocks
x wais-1-0-10/man/ebcdic2ascii.1, 450 bytes, 1 tape blocks
x wais-1-0-10/man/rmnl.1, 557 bytes, 2 tape blocks
x wais-1-0-10/man/tmpl.1, 1486 bytes, 3 tape blocks
x wais-1-0-10/man/waisindex.1, 4103 bytes, 9 tape blocks
x wais-1-0-10/man/waislookup.1, 2525 bytes, 5 tape blocks
x wais-1-0-10/man/waisparse.1, 7345 bytes, 15 tape blocks
x wais-1-0-10/man/waisreporter.1, 1489 bytes, 3 tape blocks
x wais-1-0-10/man/waissearch.1, 2771 bytes, 6 tape blocks
x wais-1-0-10/man/waisserver.1, 3723 bytes, 8 tape blocks
x wais-1-0-10/sample/README, 3335 bytes, 7 tape blocks
x wais-1-0-10/sample/data/resumes/res3.txt, 4102 bytes, 9 tape blocks
x wais-1-0-10/sample/data/resumes/README, 5243 bytes, 11 tape blocks
x wais-1-0-10/sample/data/resumes/res1.txt, 3070 bytes, 6 tape blocks
x wais-1-0-10/sample/data/resumes/res2.txt, 4634 bytes, 10 tape blocks
x wais-1-0-10/sample/data/mail/rmail, 5760 bytes, 12 tape blocks
x wais-1-0-10/sample/data/mail/README, 4394 bytes, 9 tape blocks
x wais-1-0-10/sample/data/jargon/README, 4292 bytes, 9 tape blocks
x wais-1-0-10/sample/data/jargon/convert.c, 1874 bytes, 4 tape blocks
x wais-1-0-10/sample/data/jargon/jargon.231.text, 558266 bytes, 1091 tape blocks
x wais-1-0-10/sample/data/weather/ACUS1.DOC, 4961 bytes, 10 tape blocks
x wais-1-0-10/sample/data/weather/README, 2866 bytes, 6 tape blocks
x wais-1-0-10/sample/data/weather/WXKEY.DOC, 31 bytes, 1 tape blocks
x wais-1-0-10/sample/data/weather/WXMAPARS.DOC, 36 bytes, 1 tape blocks
x wais-1-0-10/sample/data/weather/ACUS1.GIF, 13581 bytes, 27 tape blocks
x wais-1-0-10/sample/data/weather/SELSLOG.GIF, 13870 bytes, 28 tape blocks
x wais-1-0-10/sample/data/weather/WXKEY.GIF, 18865 bytes, 37 tape blocks
x wais-1-0-10/sample/data/weather/WXMAPARS.GIF, 35661 bytes, 70 tape blocks
x wais-1-0-10/sample/data/weather/SELSLOG.DOC, 5216 bytes, 11 tape blocks
x wais-1-0-10/sample/Makefile, 1327 bytes, 3 tape blocks
x wais-1-0-10/INSTALL, 11796 bytes, 24 tape blocks
x wais-1-0-10/README, 1916 bytes, 4 tape blocks
```

Also note that in this example, the current WAIS software release is wais-1-0-10. The actual release which you are unpacking may have a different name, so whenever you are asked to type wais-1-0-10, replace it with the name of the version you are using.

# Preparing for Upgrades and Releases

Before we examine the release itself, let's take a look at how WAIS releases are organized. This is just a suggested organization, if your site has different requirements, feel free to modify this structure as you see fit.

As it is shipped, each WAIS release fits completely into one directory structure. The name and version number of the release is encoded in the name of the directory. In our example, the name is wais-1-0-10. As future releases are made, the version number will be incremented. This way, you can have several versions of WAIS available on your system at the same time. Furthermore, you don't need to worry about overwriting old releases when you upgrade.

When you set up your WAIS server and WAIS databases, you won't want to encode the version number in each configuration file, since that would mean reconfiguring each file when a new release is installed. Instead we recommend making a symbolic link which maps the name of the release directory into a standard name used by the configuration files. In our example, we'll use "current" as the standard name.

Make sure you are in the top level WAIS directory, and type:

```
% cd /wais
```

and make a link

```
% ln -s wais-1-0-10 current
```

When a new release is installed (e.g. wais-1-0-11), you can switch over to the new software by removing the old link, and making a new one.

```
% rm current
% ln -s wais-1-0-11 current
```

# Installing the WAIS Programs & Online Manuals

To make the WAIS programs available to you on the command line without having to type the full pathname of each program, you will need modify your **PATH** environment variable.[1] To make the online reference manuals accessible through the UNIX **man** command, you will also need to change the **MANPATH** environment variable. It is advisable to do this after making the link described in the previous section, so that you don't have to do it again every time new software is installed. To modify these environment variables, you will need to edit your .cshrc file, which is found in your home directory. Add the following line to the bottom of the file:

```
setenv PATH /wais/current/bin:$PATH
setenv MANPATH /wais/current/man:$MANPATH
```

Make sure the .cshrc file is saved, then use the following command to load the new version into your UNIX shell:

```
% source ~/.cshrc
```

You can then try using one of the WAIS programs by typing:

```
% waisserver
USAGE: waisserver
         [-port] (the tcp-ip port, defaults to 210)
         [-directory] (use directory as the source of databases)
         [-e logFile] (the file to write log info to)
         [-u user] (if started as root, setuid to user after startup)
         [-v] (print the version of the software)
```

You can then type,

```
% man waisserver
```

to view the manual page on **waisserver**. For your convenience, the UNIX manual pages are reprinted in Chapter 8, the Reference Section.

If you want other users to be able to use the WAIS programs and manuals too, ask your system administrator to include the **setenv** command in your site's shared .cshrc file.

---

[1]The installation described applies to C shell (csh) users only. If you use another shell (eg. borne shell, korn shell, etc), you will need to modify installation to suit your shell.

# 3

# The WAIS Directory Structure

This chapter is a guided tour of the WAIS release structure. It will give you a feel for where the software components can be found, and it will serve as an introduction to administering the WAIS system as a whole. The directory structures are simply our recommendations. Feel free to adjust them to meet your own needs. If you are not sure what your needs are, don't worry, it's easy to set up a system in the default configuration, then change it around once you understand your unique constraints.

There are five main components in a WAIS installation. Each WAIS component is located in a separate directory under the root directory. In our example installation, we have selected **/wais** as our root directory.

### /wais/current

This directory contains the most recent release of the WAIS programs, the WAIS documentation, the UNIX-formatted manual pages of the WAIS programs, some sample databases to play with, and utilities. The contents of this directory is supplied for you with each new release.

### /wais/data

This directory consists of your original data which is supplied and maintained by you. If your WAIS server will be handling multiple databases, this directory is typically organized as a set of subdirectories, one for each data set.

### /wais/indexes

This directory contains WAIS indexes. The indexes help the WAIS server to quickly search and retrieve documents from your databases. For each database in the **data** directory, a corresponding index directory is created and maintained by you using. Thus this directory mirrors the subdirectory structure of the **data** directory.

### /wais/logs

Your WAIS server automatically records all client transactions in the log files of this directory. This information can be used for billing and statistical analysis.

### /wais/remote-dbs

This directory is made up of the source definitions of public WAIS databases. It is used by the get-remote-dbs script in the /wais/current/utilities directory. The script maintains a local copy of the Directory of Servers. This is only applicable if you periodically run the script and have a connection to the Internet.

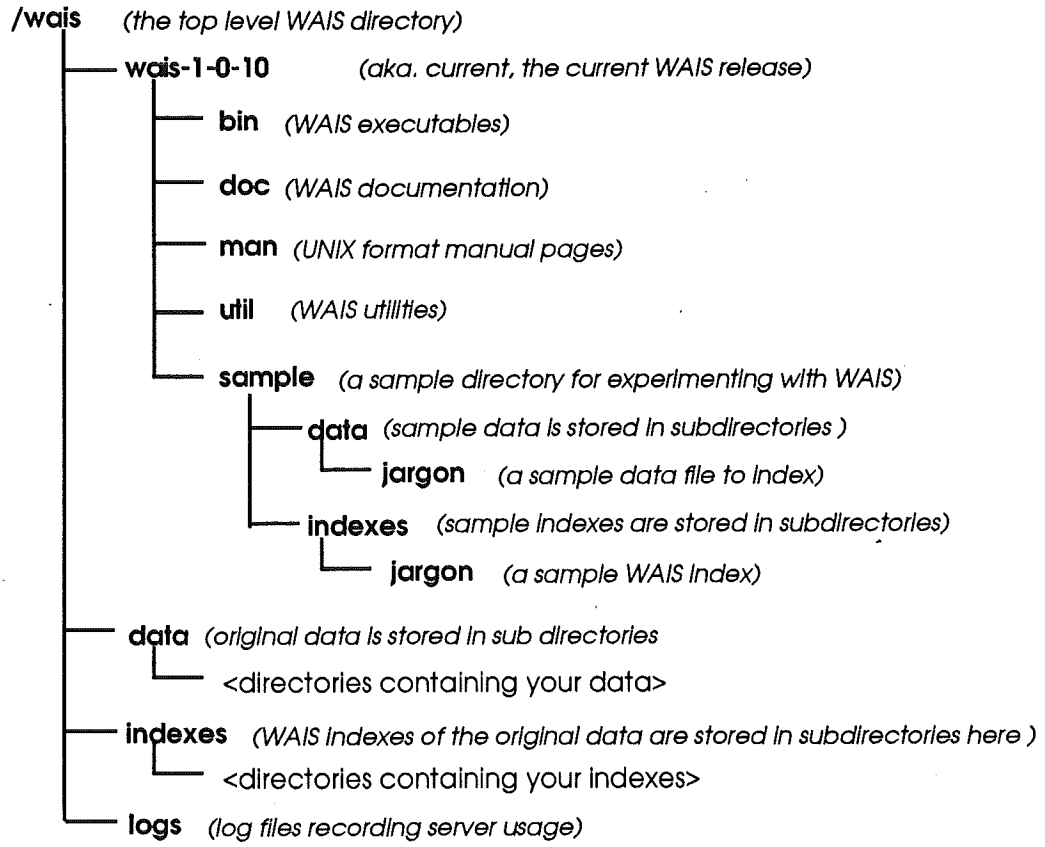


*Figure 2: The WAIS Directory Structure*

This chapter describes these directories in detail. While working through this chapter, refer to Figure 2 for a close-up view of the overall structure.

# The current Directory

The **/wais/current** directory has the WAIS software installation. It contains four subdirectories:

**bin**

This directory contains the six WAIS programs: **waisindex**, **waislookup**, **waisparse**, **waisreporter**, **waissearch**, and **waisserver** (or **waisworkstation**).

**doc**

This directory contains the WAIS Server and WAIS Workstation for UNIX Administrator's Manual, and the WAIS Client for Macintosh User's Manual.

**man**

This directory contains the UNIX-formatted manual pages for each of the WAIS programs located in the **bin** directory.

**util**

The utility directory contains programs for file formats conversions and shell scripts. See Chapter 8, the Reference Section, for the manual pages of these utility programs.

# The data Directory

The **/wais/data** directory is the home of your original data collection. You can store these files anywhere, but if they are moved or modified, the index will need to be rebuilt.[2] Files that are already part of your system need not be moved, but if you are collecting a new batch of files for indexing, it's convenient to have a standard place to put them.[3] Our convention is to collect the data files in the data subdirectory of the main WAIS root directory (e.g. /wais/data). If this is the first time you are installing WAIS, this directory will not exist yet. You can create it with the following commands.

```
% cd /wais
% mkdir data
```

Within the data directory, we build a subdirectory for each set of data files we will index. For example, if we had a set of files containing information on customer contacts, we would make a special directory for it. It could be called contacts

---

[2]Even after indexing, the original data files are critical to the functioning of a database. These are the files from which data is retrieved when a user requests a document. Therefore they must not be moved or modified without an accompanying index rebuild. Furthermore, their names must stay the same so that the server find them.

[3]The contents of /wais/data can be large, and may not fit on one disk. To solve this, you may put the data on another disk, and make a symbolic link to the files.

(/wais/data/contacts). We could also have a directory for our staff mailing list (/wais/data/staff), and another for our company's proposals (/wais/data/proposals).

# The indexes Directory

Just as the data is usually stored in it's own directory, the indexes are also stored in their own directory. Besides convenience, this organization makes serving the data easier (for more information, see the Chapter 5, Setting Up a WAIS Server). In our examples we will call the directory /wais/indexes. If this is the first time you are installing WAIS, the indexes directory will not exist yet. You can create it with the following commands.

```
% cd /wais
% mkdir indexes
```

Within the /wais/indexes directory, there should be one subdirectory for each database that you have indexed. Each subdirectory has the same name as the database. The contents of the index directory are explained in detail in Chapter 4, Building a WAIS Database.

# The logs Directory

The server logs should all be kept in a single directory. We suggest **/wais/logs**. If this is the first time you are installing WAIS, the log directory will not exist yet. You can create it with the following commands.

```
% cd /wais
% mkdir logs
```

The server records entries in the log file located in this directory. Chapter 7, Monitoring WAIS Usage, will introduce automated scripts which keep a separate log file for each day, and send a daily usage report to the system administrator.

# 4

# Building a WAIS Database

## What is a WAIS Database?

A WAIS database is made up of two main components: a collection of information, and a WAIS index of this collection. The collection of information is generally referred to as the original data, or just data. It is supplied by you. The WAIS index is a set of files generated by the WAIS programs that facilitate fast search and retrieval of the information stored in the database. Taken together, the database and the WAIS index are the essential ingredients making up a complete WAIS database system.

Typically, the original data is made up of a set of documents, headlines, and words. A document is the smallest retrievable element of the database. For example, a database may contain a volume of journals, where each article of the journal is as separate document. Another example is electronic mail, where each mail message is a different document. A document may also be made up of text, images, or video, or a combination thereof. In a database of resumes, for instance, each document may contain a resume of a person in your company's technical marketing division, a picture of that person, and possibly even a short video of a marketing presentation that the person once gave.

Each document is associated with a headline. A headline is one or more words that embody the main idea behind the document. When a client sends a question to a server, the server responds with a list of headlines that represent the most relevant documents related to the client's question. Generally, each headline is automatically extracted for you from the database.

In addition to a headline, each document is also associated with one or more words. These words constitute the text of the document.

Together, the headline and the words are used to determine how relevant a document is to a client's question.

A WAIS database is constructed by using the WAIS parser and WAIS indexer programs to create a WAIS Index. The parser takes the original data and separates it into documents, headlines, and words. The indexer takes the information generated by the parser and creates the WAIS Index. Next, the WAIS lookup program is used to check the search and retrieval of information in the database.

# What's in a WAIS Index?

A WAIS index is made up of several files. This section describes each of these files and explains how they are used for the search and retrieval of documents in a database. Unless otherwise noted, index files are binary files, and can not be edited by hand.

### Source Description (.src)

The Source Description File describes the database and the server. It is user by the client to contact the server and search the database. Source description files are distributed to clients by the directory-of-servers. They are ASCII files, and can be edited by hand.

### Access List (.acc)

The Access List contains the addresses of all machines which are allowed to search the database. They are editable files that you create, and are described in detail in Chapter 5, Setting Up a WAIS Server.

### Dictionary (.dct)

The Dictionary contains a list of all the words used in a database.

### Inverted File (.inv)

The Inverted File contains a list of all the words in the database, and for each word, it lists all the documents which contain that word.

### Document Table (.doc)

The Document Table contains a record of each document in the database.

### Headline Table (.hl)

The Headline Table contains headlines of all the documents in the database.

### Filename Table (.fn)

The Filename Table contains a list of the filenames of the original data files.

## Catalog (.cat)

The Catalog contains a human readable list of headlines and document identifiers for some or all of the documents in the database. This list may be returned to a user whose search has gone poorly, as an aid to help them understand the contents of the database.

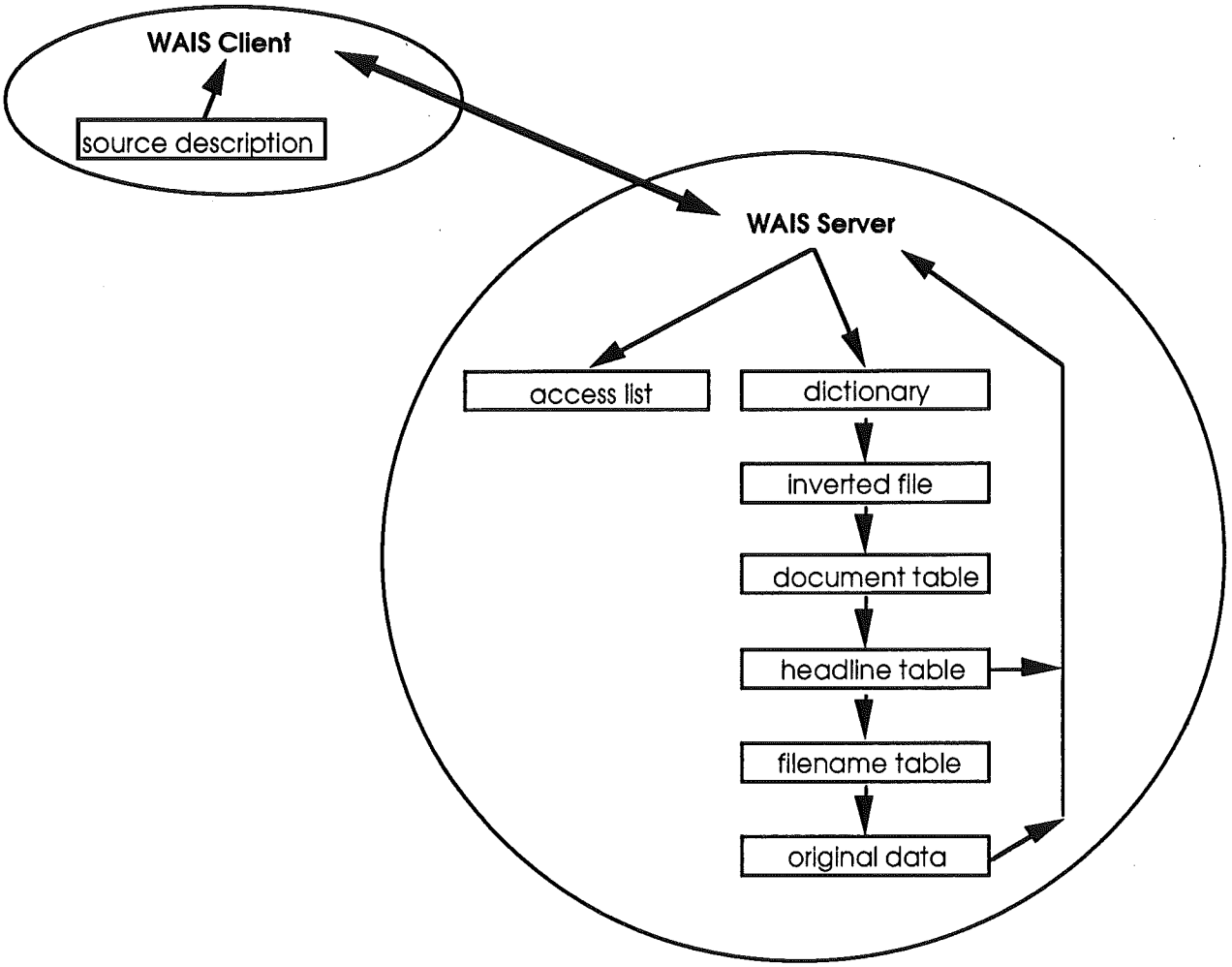


*Figure 3: How index files are used during a search.*

The interaction between the WAIS index files is illustrated in Figure 3. A client uses a source description file to contact the server. The server checks the access list to make sure the client has permission to access this database. If so, it looks up the words in the client's question in the database's dictionary. It uses information in the dictionary to look up lists of documents in the inverted file. The document table is then consulted, which gives a pointer to the headline table, which in turn gives a pointer to the filename table. Finally, using information from the filename table, the original data file, is consulted, and results are returned to the client.

# Items to Consider

Before building a WAIS database, there are several items that you will need to consider.

- How much disk space your original data requires, and how much additional space you will need to perform indexing and to store your WAIS index.
- The best location for your WAIS index. This decision consists of selecting the directory for your index, and the server machine on which the index will be stored.
- Where the documents, headlines, and words exist in relationship to the files in your database. This will determine the selection of the parse format for the files in your database.
- The format your data should be displayed in to a client. Examples of display formats include Microsoft Word and GIF Images.

Each of these issues is addressed in detail in the following sections.

# Disk Space Requirements

To build a WAIS index for your database, you will need to ensure that you have enough free disk space to build the indexes and to store the resulting index files. This section provides you with a means for roughly calculating your storage requirements. The calculation is broken into three parts: storage of the original data. storage during the process of building the index files, and storage of the index files.

First, you will need to measure how many megabytes of storage your original data requires. To do this, use the **du -s** command on your database directory to summarize how much disk space your database currently requires.

```
% cd
% du -s /wais/data/mydata
534073
```

In this example, approximately 534 megabytes of disk space is required to store the data in the directory called /wais/data/mydata.

To generalize the calculation, let us assume that the amount of disk space required is *n* megabytes.

Next, you will need to determine how much space is required during the creation of the index files. In general, building an index for *n* megabytes of data requires an additional *n* megabytes of disk space. Half of this space is necessary for temporary storage during the building process, and the other half is required to store the finished index files.

After the index files have been built, the amount of disk space required to store the index files is 1/3 to 1/2 of *n*. Thus, the total amount of disk space required to store a full WAIS database is at most *1.5n*. This corresponds to the sum of the amount required to store the original data, *n*, plus the amount required to store the index files, which is less than *0.5n*.

# Index Location

When deciding on the location of the index files, there are two issues to consider. The first issue concerns the directory in which the indexes are stored, and the second issue concerns the machine on which these indexes physically reside. Both issues are discussed in detail below, with our recommendations on how you can best address these issues at your site.

For public or shared databases, we recommend storing the indexes in one common directory. As described in Chapter 3, The WAIS Directory Structure, the suggested location of this directory is /wais/indexes. If your indexes are stored on different disks, you can use symbolic links make it appear that the indexes are located in the same directory.

The reason for placing the indexes in a common directory is so that clients requesting information from a database do not have to know the full pathname of the index for that database. When the server is set up as described in Chapter 5, Setting Up a WAIS Server, this common directory is specified with the **-d** switch. When a client requests information from a database on the server, only the name of the database is required and not the full pathname. This allows

you to change the location of the index directory without having to modify the clients. Instead, you only need to update the server's **-d** switch.

Within the /wais/indexes directory, there should be one directory for each database that you index. Index files may be moved around, as long as all the files for a given database are in a single directory, and a link to the directory is maintained in /wais/indexes. This means that indexes can be created on disks other than the one containing the WAIS software. This is very important when building large databases.

It is highly recommended that indexes be physically located on the disks of the server machine. This strategy is recommended primarily for performance reasons. When a client sends a request to a server machine, the server searches through the index files to determine which documents are most relevant to the information request. This process is most efficient if the index files are local to the server.

# Parse Format

The parse format tells the WAIS parser how each file in your database is organized. More specifically, the parse format specifies how the parser should distinguish each document in your database. For each document, it also specifies how to determine the headline and content-bearing words. For example, the parse format determines if a file contains a single document, or a set of documents. It also determines if the headline is the filename, or a string embedded in the document.

A parse format which best matches your database should be selected from the formats described in this section. In most cases, the format required by your database will be a supported format. In other cases, you may need to reformat your data to match an existing format, or build your own parser.

**dash**

The **dash** parse format is useful if a single file contains multiple distinct documents. In the dash format, each document is separated by a row containing a minimum of 20 dash characters, "-". The line

following the dashed line is expected to contain a headline, followed by the text of the document.

### dvi

The **dvi** parse format is for Device Independent Printer Output files. The filename is used for the headline, and the contents of the file supply the words of the document.

### filename

The **filename** parse format treats each file as a single document, and uses the filename as the headline. The contents of the file however are NOT parsed unless the **-contents** switch is present. This format is useful for databases constructed of many individual binary files, for example, whose contents are not words.

### first-line

The **first-line** parse format specifies that each file contains a single document, and that the first non-blank line of the file is the headline, and the remainder of the file is parsed as words.

### first-words

The **first-words** parse format is similar to first-line, except that the headline is the first 100 non-whitespace characters in the file.

### gif

The **gif** parse format is for Graphics Interchange Format files. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such as gif, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

### mail-digest

The **mail-digest** parse format is for standard Internet mail digest files. A mail-digest file contains one or more electronic mail messages, in which each mail message is parsed as a separate document. The subject line of the mail message is the headline, and the body of the message contains the words of the document.

### mail-or-rmail

The **mail-or-rmail** parse format is used for UNIX mail files. A mail file is a single file containing one or more electronic mail messages, where each mail message is parsed as a separate document. The subject line of the mail message is the headline, and the body of the message contains the words of the document. In addition, the sender, the receiver, and the date are recognized as field information.

### netnews

The **netnews** parse format is used for Internet Network News, where each Network News or Read News file contains one or more news messages. Each news message is parsed as a document, where the subject line is the headline, and the body of the message contains the words of the document.

### one-line

The **one-line** parse format is a simple format that treats each line of a file as a separate document. The line also forms the headline for that document.

### paragraph

Like the dash format, the **paragraph** parse format is useful if a single file contains multiple distinct documents, or paragraphs. In the paragraph format, each paragraph is separated by one or more blank lines. The first line of each paragraph is the headline which is followed by the text of the document.

### pict

The **pict** parse format is for Apple PICT image file formats. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such as pict, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

### ps

The **ps** parse format is for PostScript files. The filename is used for the headline, and the contents of the file supply the words of the document.

### source

The **source** file format (.src) is a file format generated by the WAIS indexer for the Directory of Servers. The file typically contains information about the database, and is parsed exactly like the text file format.

### text

In **text** format, each file is treated as a single document, the filename is used as the headline, and the contents of the file is parsed as words. This format is useful for databases constructed of many individual files. This is the default parse format.

### tiff

The **tiff** parse format is for tagged interchange file formats. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such tiff, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

The display format determines how a client should display a retrieved document. In many cases, the document is a simple text file, and thus has no special display needs. In other cases, the document may be in a specific format that should be displayed with a special display program. For example, suppose a document was generated using Microsoft Word. The client that retrieves this document must be told that the document is in Microsoft Word

format in order to display it correctly. To do this, a display format of **MS-WORD** is given to the parser.

The parser associates a display format with each document. The parser passes the display format through to the indexer, which in turn stores it for later use by the server. When a client retrieves a document, the server sends the client both the document and its display format. It's up to the client to decide whether or not it can display this format.

Examples of display formats supported on many existing WAIS clients are described in Table 1.

| Display Format | Description of Format |
|---|---|
| DVI | Device-Independent Printer Output |
| GIF | Graphics Interchange Format<br>CompuServe Images |
| MIME | AT&T Multimedia Document |
| MS-EXCEL | Microsoft Excel Spreadsheet |
| MS-POWERPOINT | Microsoft PowerPoint Slides |
| MS-WORD | Microsoft Word Document |
| PERSUASION | Aldus Persuasion |
| PICT | Apple PICT Image |
| PS | PostScript |
| QUICKTIME | Apple Quicktime Movie |
| TEXT | ASCII Text |
| TEXT-FTP | Special FTP File Format |
| TIFF | Tagged Interchange File Format (.tif)<br>A Universal Raster Image Format |
| WQST | WAIS Question Format (.qst) |
| WSRC | WAIS Source Format (.src) |

*Table 1: Display Formats*

The exact name of the display format you select depends on the display format supported by the WAIS clients that will be accessing your database. For this reason, you may want to check with the client programs to determine what display formats it supports.

If a display format is not specified, the default value of the display format is dependent on what is specified for the parse format. If the parse format is **gif**, for example, then the display format defaults to **GIF**, respectively. Otherwise, the default display format is TEXT.

# Building an Index

A WAIS index is created from your original collection of data using the **waisparse** and **waisindex** programs. **waisparse** takes a collection of documents in the form of files and directories, and separates the data into a stream of documents. For each document, the parser identifies a headline and the content-bearing words. **waisindex** takes a stream of documents from **waisparse** and creates an index for fast search and retrieval of these documents. This completes the building of your WAIS database.

To build a WAIS index, use the **waisparse** command piped into the **waisindex** command. The command line is:

```
waisparse -parse parse-format
          -display display-format list-of-filenames |
          waisindex -d database-name
```

where *parse-format* and *display-format* are the parse and display formats of your data, respectively, *list-of-filenames* is a list of the files of your data, and *database-name* is the pathname where the index files are to be stored. If the directory specified by *database-name* does not exist, the indexer creates it. If it already exists, the indexer overwrites all index files, except for the .acc and .src files.

## Single-Document Files

In the simplest case, a database may consist of one file per document, where the headline of the each document is the same as the filename. An example usage looks like the following::

```
% waisparse /wais/data/resume/*.txt |
            waisindex -d /wais/indexes/resume
```

Note that the default parse and display formats are **text** and **TEXT**, respectively. As another example, consider indexing a set of C source code files. In this case, programmers can use the WAIS system to quickly search through large amounts of source code spread across multiple development directories. As above, each source code file is considered a single document.

```
% waisparse -r /dev/src/*.[ch] |
            waisindex -d /wais/indexes/source-code
```

The optional **-r** switch tells the parser to recursively descend the /dev/src directory, looking for files with either a **.c** or **.h** extension.

Consider another example where the filenames of the database are listed in a file. For example,

```
% cat filelist | waisparse - |
                 waisindex -d /wais/indexes/mydatabase
```

where **filelist** is the name of the file containing a list of the files in your database. The - tells **waisparse** to read its arguments from the standard input. Additional database files may be specified after the - character.

### A Multi-Document File

In the case where all documents are contained in a single file, only a single filename needs to be specified in the call to **waisparse**. The following is an example of how to index a personal electronic mail file:

```
% waisparse -parse mail-or-rmail /usr/smith/RMAIL |
            waisindex -d /usr/smith/indexes
```

Note that this example did not use the /wais/indexes for the **-d** switch to **waisindex**. This is important if you wish to maintain your own personal WAIS database.

### Multiple Parse Formats

If your database consists of multiple parse formats, you can specify each format in the command line. For example, consider a database of resumes, where a subset of the resumes have the headline (name) as the filename, and another subset are in Microsoft Word format. The command line might look like the following:

```
% waisparse -parse first-line
            /wais/data/resume/*.txt
            -parse text -display MS-WORD
            /wais/data/resume/*.doc |
            waisindex -d wais/indexes/resume
```

The first set of resumes have a parse format of **first-line** and a default display format of **TEXT**, and the second set of resumes have a parse format of **text** and a display format of **MS-WORD**. Each time a new **-parse** switch is encountered, the display format is reset to the default value, **TEXT**. Thus, in the event that your display format is different from the default value, it is important to list the parse format, before specifying the display format.

### Multi-Display Documents

A single document may be made up of multiple files, where each file has a different display format. One of the files is designated as the primary file. It contains the headline and the words that are parsed and indexed. The other files of the document are called the supporting files, and are specified with the **-assoc** switch to **waisparse.**

For instance, you may want to associate an image file with a textual description file. When a client retrieves the document, the client may give the user the choice of viewing the image display format, the text format, or both. The command line for this might look like:

```
% waisparse -parse first-line
            -assoc tif TIFF
            /wais/data/resume/*.txt |
            waisindex -d /wais/indexes/resume
```

Here the **-assoc** switch is used to associate a TIFF image, stored in files with a .tif extension, with a text file, where the latter files are designated as the primary files. For example, the primary file /wais/data/resume/foo.txt is matched with the image file /wais/data/resume/foo.tif.

Another example where multiple display formats are useful is where a document already exists in multiple formats. A technical report, for example, may be available in text, Postscript, and Microsoft Word format where you want all three formats available to the user.

```
% waisparse -assoc ps PS -assoc doc MS-WORD
            /wais/data/resume/*.txt |
            waisindex -d /wais/indexes/resume
```

The files that are parsed and indexed are specified as /wais/data/resume/*.txt using the default **text** parse format. The Postscript and Microsoft Word format files, /wais/data/resume/*.ps and /wais/data/resume/*.doc, respectively, are associated with these text files. The display formats are PS and MS-WORD.

### WAIS Server Documents

A directory of servers is a WAIS database made up of .src files. A directory-of-servers database helps users find out about other WAIS databases. You can create your own directory-of-servers database by parsing and indexing the .src files in your databases index

directories. This could be performed with the following command line:

```
% waisparse -parse server /wais/indexes/*/*.src |
        waisindex -d /wais/indexes/directory-of-servers
```

# Testing and Troubleshooting

The **waislookup** program is an interactive interpreter for performing search, retrieval, and relevance feedback directly on a database, without using the server. It is used to test and troubleshoot WAIS indexes. To test a database using **waislookup**, use the command line:

**waislookup -d** *database-name*

where *database-name* is the same directory name that you specified in the **-d** switch to **waisindex**. If a search fails, check for errors from **waisparse** or **waisindex**.

# Maintenance

## Creating Additional Databases

Creating additional WAIS databases as easy as creating your first database. All you need to do is to follow the same procedure as for creating the first database. Furthermore, the server does not require any modifications. If you want to make the database publicly accessible, you may need to reindex your local directory of servers database, or notify the public Directory of Servers.

## Moving a Database

Moving a database involves two main issues: moving the original data, and moving the WAIS Index. To move the original data, reindexing is required. If a database is moved without reindexing, use of the database will always result in no documents since the server will be unable to find any documents. The log files will record "attempt to retrieve data for missing document".

Moving a WAIS index does not require reindexing. All that is required is renaming the directory. If the database is part of a directory of servers database, you will need to reindex that database.

In any event, you will also need to check that the permissions for accessing the new location are still valid for the original data and for the WAIS index.

## Appending to a Database

> Note: This feature will be available in the upcoming bug-fix release.

Suppose you would like to add more documents to your database without having to reindex. You can do this with the optional **-a** switch to **waisindex** as follows:

```
waisparse filelist | waisindex -a -d database-name
```

This command line parses the list of files specified by *filelist* and tells the indexer to append the documents to the index files. If one of the listed files had already been indexed, and the file has not been changed, the file is not appended. If the modification date is different, the new file is appended to the index, and the old file is marked as removed.

If you need to append to your database on a periodic basis, this procedure can be wrapped into a shell script and the UNIX **cron** facility can be used to periodically execute the script.

# 5

# Setting Up a WAIS Server

## What is a WAIS Server?

The WAIS server is a process that runs on a machine containing one or more WAIS databases, and services requests from client processes. The operation of the server is very simple. First a WAIS client initiates a request to a server machine. Next, the server machine responds by creating a new server process to handle the client's request.

Once created, the server process reads the client's question, performs a search on the requested databases, and returns a relevance-ranked list of document headlines to the client. If the user wants to view one of the documents, a retrieval request is sent from the client to the server. The server verifies that the document requested is in the database, and if so, retrieves the data from the original file that was indexed. Finally, when the client has completed all requests from the database, the server process terminates.

## Items to Consider

Before setting up your WAIS server, you will need to decide on the items described below. Where appropriate, we have also included our recommendations for how to approach each item.

- The port number on which to run the server. This is a number that is agreed upon between the WAIS server and WAIS client programs that wish to communicate with it. It is common practice for public WAIS servers to use port number 210.
- The root directory where the indexes of your server's databases are located. The suggested location is in a directory named /wais/indexes.

- The name and location of the log files. The recommended location is in the file named /wais/logs/server.log.
- The user that will run the server. Typically, this is the superuser, the user named root.
- The mode the server will run in, either standalone mode or daemon mode. In practice, standalone mode is only used for debugging purposes, and daemon mode is recommended for long term use. Setting up the server for standalone and daemon mode are described in this chapter.
- The Internet addresses (IP address) of those machines having access to your server's databases. Setting up restricted access lists for these machines is detailed later in this chapter.

To set up your WAIS server, we begin first by describing how to run your server in standalone mode. Once you feel confident that the server has been correctly set up in standalone mode, we then present how to set it up in daemon mode for long-term usage. And finally, we describe how to set up an access security list to provide restricted access to each of your databases.

# Standalone vs. Daemon Mode

For each new client requesting services, the responsibility for creating a new server process depends on whether the server is being run in standalone or daemon mode. In standalone mode, a WAIS server daemon process is always in existence and continually running in the background. For each new client requesting service, this daemon process forks off a new child server process to handle the client.

In daemon mode, the network daemon **inetd** is responsible for creating a new server process for each new client requesting servicing. The **inetd** daemon is the Internet network daemon that manages many network services according to the configuration specified by the /etc/inetd.conf file. The **inetd** daemon runs in the background and manages client requests by spawning off a child server process to service these requests.

We suggest that you initially try running the server in standalone mode because the server is much easier to test in standalone mode

than daemon mode. In standalone mode, the server is started directly from the shell. This output allows you to easily detect if anything goes wrong with the server. On the other hand, in daemon mode there is much less feedback in the event of an error, and it is thus more difficult to debug.

Once the server has been tested in standalone mode, the WAIS server can then be set up to run as an new network service of the **inetd** daemon. Running the server in daemon mode eliminates the need to have a separate server process to spawn off child servers for each new client process. Instead, the **inetd** daemon services these requests itself by forking new child servers, one for each new client.

# Standalone Mode

To run the WAIS server in standalone mode, you can invoke the **waisserver** (or **waisworkstation**) program either at a UNIX prompt, or through a shell script. To run the program, use the following command:

```
waisserver -p port -d directory
```

where *port* is the port number through which the server communicates with the client, and *directory* is the root location of the database index files.

If the port number is less than 1024, the command must be run as superuser. It is common practice to use port 210 for public WAIS servers, but for testing purposes, you may want to choose a different port number that does not require you to be superuser. If you choose to select a port number other than 210, first check the /etc/services file to ensure that this port number is not being used by any other network service.

An example usage,

```
% waisserver -p 8000 -d /wais/indexes &
1: 0: Jan 5 05:09:43 1904: 100: running server WAIS Inc
```

selects port number 8000. The **&** at the end of the command line tells the shell to run the command in the background. This allows you to type other commands at the prompt while the server process

is running. Once you have started the server, check to see that the server process has started:

```
% ps -auxww | grep waisserver
margaret     3207   0.0  2.2 2.16M  360K p4 S     0:00 waisserver -p...
margaret     3212   0.0  1.3 1.52M  208K p4 S     0:00 grep waisserver
```

In this example, the server's process identification (PID) number is 3207. You may want to keep track of the PID number in case you need to kill the process.

And for the final step in the testing process, you may want to run **waissearch** to verify that the data in each of your databases is accessible through the server. This is described in the section, Testing Your Server.

Once you have completed and verified that the server runs in standalone mode, you may then want to kill your server process and try running the server in daemon mode. For the **kill** command, use the PID number that you recorded from running the **ps** command.

```
% kill 3207
2: Terminated waisserver -p 8000 -d /wais/indexes...
```

If you decide to continue to run the server in standalone mode, keep in mind that the server will need to be restarted every time the machine is rebooted.

# Daemon Mode

To run the server in daemon mode, you must modify the /etc/inetd.conf and /etc/services files as superuser. A sample entry for setting up the server in the /etc/inetd.conf file follows:

```
z3950 stream tcp nowait root /wais/current/bin/waisserver waisserver.d
-e /wais/logs/server.log -d /wais/indexes
```

The entry must be specified as a single line in the /etc/inetd.conf file (no carriage returns). Each item in this entry is described below:

### z3950

The service name is z3950. The service name is mainly an alias for the port number associated with this service. It should be listed in the /etc/services file as described later in this section.

**stream**
The socket type is stream.

**tcp**
The protocol is TCP.

**nowait**
After starting up a new server process, nowait specifies that the **inetd** daemon should not wait for it to complete before servicing another request from a new client.

**root**
The user running the server program is root.

**/wais/current/bin/waisserver**
This is the name of the command to run to invoke the server. This should be set to the complete pathname of the **waisserver** executable at your site.

**waisserver.d**
When a server program is activated by the **inetd** daemon, the name of the server program is called **waisserver.d**.

**-e /wais/logs/server.log**
The **-e** argument to the **waisserver** program specifies the filename for where the server should log its information. See Chapter 7, Monitoring WAIS Usage, for more information on the log file and generating WAIS Usage Reports.

**-d /wais/indexes**
The **-d** switch to the **waisserver** program specifies the directory containing the database index files.

Additional switches may also be specified to the **waisserver** program by appending them to the **inetd** entry.

Whenever a new client tries to connect to port 210, the **inetd** daemon reads its configuration from the /etc/inetd/conf file and starts up a server program by invoking the **waisserver** command with the arguments specified in the /etc/inetd.conf entry.

The next step required for daemon mode setup is to modify the /etc/services file. The new entry to this file looks like the following:

```
z3950 210/tcp # Wide Area Information Server (WAIS)
```

This entry tells the **inetd** daemon what port number and protocol to use for the service defined in the /etc/inetd.conf file. Each item in this entry is:

**z3950**
This is the same service name as specified in the /etc/inetd.conf file.

**210**
The port number that a client process should use to talk to the server is port 210. This is the standard port number for public WAIS clients and servers.

**tcp**
The protocol to use for communicating is **tcp**.

**# Wide Area Information Server (WAIS)**
Anything after a # character is interpreted as a comment. This comment let other system administrators know about this service and corresponding port number.

In order for your modification to the **inetd** daemon to take effect, you will need to restart the daemon. This can be done by rebooting, or by finding the **inetd** daemon PID number with **ps** and sending a HUP signal to the process.

```
% ps -auxww | grep inetd
root     136 80 330 17 ?     S  Jan 5 0:04 /usr/sbin/inetd -s
morris 2734 12 135 95 pts/2 S    00:31:45 grep inetd
% kill -HUP 136
```

You may also want to update your NIS or NIS+ server.

Once you have modified the /etc/inetd.conf and /etc/services files and have updates them, you are ready to test the WAIS server in daemon mode. Like in the standalone mode, try performing a search and retrieval using the **waissearch** client.

When the server is running in daemon mode, it is automatically started whenever a client connects to it. You do not need to do anything special, even after rebooting the machine.

# Testing Your Server

The WAIS search program is mainly used for testing and troubleshooting WAIS servers. To test a database using WAIS search, use the following command line:

**waissearch -h** *host-machine* **-p** *service-or-port* **-d** *database-name word...*

With this command, you should be able to successfully run queries. If you are not able to run queries, there are several possible causes. Check the **inetd** entry, making sure that the files exist, permissions are correctly set, and the user you selected exists. Check that **inetd** has been restarted, and reboot if necessary. Make sure the machine you are using is capable of communicating with the machine the server is running on. Finally, try running the server in standalone mode, with the same arguments you used in the inetd.conf.

Once you get this working, try connecting using the Macintosh or other WAIS clients.

# Setting Up Security

The WAIS server uses an access-list security system to limit client access to WAIS databases. Before processing a client's request, the server checks to see if the requesting client has access to the requested database. It does this by checking an access list maintained for each database. The access list tells the server the legal client machines that have access to the database. This section describes how to set up the access list security system for your server.

For any given database, the access list is a file with a **.acc** extension and is located in the same directory as the indexes of that database. For example, if your database is named:

```
/wais/indexes/mydatabase
```

as specified by the **-d** switch to the **waisindex** program, then the pathname of the access security list file for this database should be:

```
/wais/indexes/mydatabase/index.acc
```

If this file is nonexistent, any client can access the information in the database. If this file does exist, then only those clients whose machine is listed in the file will be allowed access the database. All others will get an error message, suggesting the user contact the database maintainer.

The .acc file consists of the Internet address (IP address) of certified client machines, such as:

```
IP:+:192.216.46.104
```

The "IP:+:" must appear exactly at the beginning of a line, followed by the IP address in Internet "." notation. Comments can be entered into the file by using **#:**

```
#This is a comment
IP:+:192.216.46.104
#This is another comment
IP:+:192.216.46.104  # yet another comment
```

Wildcards, or asterisks, can be used in the IP address in replacement for any of the 4 component numbers. For example, the entry,

```
IP:+:192.216.46.*
```

allows access to machines with IP addresses 192.216.46.0 through 192.216.46.255, and the entry,

```
IP:+:192.216.*.*
```

gives access to machines with IP address from 192.216.0.0 to 192.216.255.255.

# 6

# Customizing a WAIS Database

## Customizable Stopword List

A stopword is a frequently used word that, when encountered in a user question, is ignored. For example, since the word "the" commonly appears throughout the English language, it is typically regarded as a stopword. You can specify a customized file of stopwords as an optional argument to the **waisindex** program using the **-sw** switch.

```
waisindex -d database-name -sw filename
```

where *filename* is the pathname of the file containing your customized list of stopwords. Each stopword in this list is separated by a carriage return. This procedure is general enough that a different stopword list may be specially customized for each database you maintain.

If a stopword list is not specified, the WAIS indexer uses a built-in list of approximately 300 stopwords. This default list is provided with the installation in the file /wais/current/util/stopwords.txt. For completeness, the stopword list is also included in Appendix C. You may use this file as a template for creating your own stopword list. If you do not want any stopwords, you can create one empty stopword file, or use /dev/null which is the name of an empty file on most UNIX systems.

## Fielded Search

For data collections whose documents are structured in a semi-regular format, the regular portions of the documents can be tagged by the WAIS parser as fields. A question specified by a client process can then ask the WAIS server to search on only those documents

containing a user-specified value of a particular field. Performing a restricted search based on the value of field or set of fields is called fielded search. In its simplest form, a question containing a fielded search is specified as follows:

```
field-name=field-value
```

This question tells the WAIS server to only search for documents whose field, specified by *field-name*, contains the value specified by *field-value*. The WAIS parser can support up to 254 unique fields.

The **mail-or-rmail** parse format is a good example of a parse format in which fields are tagged. For this parse format, the WAIS parser detects the **to** and **cc** fields, the **from** and **sender** fields, the **subject** field, and the **date** field. An example of a question using fielded search is:

```
to=kathy AND from=david
```

The WAIS server would then return documents that contain both a **to** field of **kathy** and a **from** field of **david**, where **AND** is a boolean operator for set intersection.

WAIS Inc develops and maintains customized parser formats for detecting specific fields in your documents. Please contact WAIS Inc for more information on developing custom parsers for fielded search.

# Stemming

Stemming is a technique used to automatically derive variations of a queried word. These variations are then used as part of the search. If a question contains the word "skate", for example, stemming is used to find documents that may also include "skates", "skated", and "skating". Two types of stemming are supported: Plural and Porter stemming. Plural stemming attempts to determine the plural form of a word. Porter stemming attempts to find the real base, or stem, of a word and derive any possible alternate variations.

You can specify the stemming algorithm as an optional argument to the **waisindex** program with the **-stem** switch as follows:

```
waisindex -d database-name -stem stemmer-name
```

where *database-name* is the pathname for the database directory, and *stemmer-name* is one of **plural** or **porter**. A different stemming algorithm may be used for each database you maintain. If this option is not specified, the default is to use no stemming. Example usages include:

```
% waisindex -d /wais/indexes/resumes -stem plural
% waisindex -d /wais/indexes/jargon -stem porter
```

# Query Report

A query report is a document created by the server that describes how a client's question is parsed by the server. When a client asks a question of a database, and if the query reporter is enabled for that server, the server creates and returns a query report to the client. The query report is the last document in the relevance-ranked list of documents returned by the server. The headline of the query report is listed as 'Query Report for this Search', and its relevance score is 1. Since the query report is an actual document, it may be retrieved for viewing by the client.

The query report contains the following information:

- The database being questioned,
- The original question,
- The boolean equivalent of the question in infix notation. This notation is a fully-parenthesized version of the question, showing the boolean operator precedence.
- The boolean equivalent of the question is displayed as a tree.
- The number of documents and the number of words in the database,
- The number of unique words in the database,
- The number of times each word in the question occurred in the database,
- The expanded search words resulting from right truncation, and
- The number of documents found that satisfied the question, and the amount of elapsed time it took to perform the search.

The purpose of this information is to give the user feedback on how the question was interpreted by the server, and on how well the

information in the database matched the words in the question. Below is an example query report generated from the following question using the AND boolean operator: "carbon monox* AND poison". Simply stated, the question is looking for documents on carbon monox* and poison*. The question uses right truncation for monox* and poison* to match words such as monoxide, monoximes, etc., and poisoned, poisoning, and poisonous.

```
Headline: Query Report for this Search
This is the search report for the search you ran on Jun  3 13:31:51 1993.
It is a temporary file, and will expire about an hour after the search.

-------------------------------------------
Searching /proj/wais/wais-sources/doe...

Your query:

  carbon monox* AND poison*

is equivalent to:

  ((carbon monox*) AND (poison*))

and was interpreted as:

  AND
   ( carbon monox*
     poison*
   )

The database contains 39,062,401 words in 230,750 documents.
There are 639,200 different words.

carbon occurs 30,404 times in 14,896 documents.
monox* is expanded to:
  monoxide occurs 3,825 times in 2,515 documents.
  monoximes occurs 1 time in 1 document.
  monoxodithioacetal occurs 1 time in 1 document.
  monoxygenases occurs 2 times in 2 documents.
  monoxyhemoglobin occurs 1 time in 1 document.
poison* is expanded to:
  poisoned occurs 61 times in 49 documents.
  poisoning occurs 486 times in 283 documents.
  poisonous occurs 17 times in 14 documents.

The search found 67 documents.  It took about 5 seconds.

-------------------------------------------

The search was performed by a WAIS Inc server:  WAIS waisserver 1.0.9i.
For more information email info@wais.com.
```

Query reporting can be enabled for a server by creating a subdirectory called **wais-tmp** in the index directory, the directory that you specified in the **-d** switch to **waisserver**. For example, if

your database directory is called /wais/indexes, create a wais-tmp directory as follows:

```
% cd /wais/indexes
% mkdir wais-tmp
```

The query reporter uses the wais-tmp directory for temporary storage of the query report. Query reports are automatically flushed after about 1 hour.

# 7

# Monitoring WAIS Usage

## What Information is Logged

The WAIS server automatically records all transactions in the log file that you specified with the **-e** switch to **waisserver**. In our example WAIS server, the pathname of the file was set to /wais/logs/server.log. Generally, the information recorded in this file is so detailed that it is difficult to obtain a good measure of the usage patterns of your databases. Instead, usage characteristics are extracted from the log file and summarized using the WAIS Reporter as described in the next section. For the curious, however, the remainder of this section is devoted to a brief overview of the information recorded in the log file.

For each client process requesting service, the WAIS server records the WAIS server daemon process ID, the current count on the number of transactions performed for this client, the date, the time, and the type of transaction. The WAIS server records six main transaction types:

- Opening a connection,
- Searching a database,
- Returning results from a search,
- Retrieving a document,
- Closing a connection, and
- Errors and warnings.

If a search transaction is performed, the server also records the name of the database and the client's question. If results were returned from a search, the number of documents found and the document identifiers are also logged. And finally, if a document is retrieved, the document identifier, the database name, the document size, and the document display format are all recorded.

# The WAIS Reporter

The WAIS Reporter can be used to summarize the information contained in the log file generated by the WAIS server. The generated summary, called the WAIS Usage Report, contains the following information:

- **Total number of connections**: This number represents the total number of independent client connections made to the WAIS server, where a connection equates to one **inetd**-spawned **waisserver** daemon process. A single connection can span over multiple searches and retrievals, and over multiple databases
- **Number of different machines connecting**: This number is the total number of client machines requesting services from this WAIS server.
- **Total number of searches**: This number is the total number of searches requested by all clients.
- **The total connect time (seconds)**: This number is the sum of the connect time of all clients. The connect time is the lifetime of each daemon server process, in seconds. The majority of the connect time is idle time.
- **The total search time (seconds)**: This number is the sum of the search time of each daemon server process, where the search time is the elapsed time, in seconds, that each daemon spends servicing its client's search request.
- **Searches returning zero hits**: This number is the total number of search requests resulting in no matches, where the daemon server process was unable to find any documents matching the client's question.
- **Total number of documents retrieved**: This number is the total number of documents retrieved by all clients.
- **Total number of db's searched**: This number is the total number of different WAIS databases that clients have searched.
- **Number of searches with no DB name**: This number is the total number of times clients requested a search without specifying the database name. When a database name is not specified in the search request, the default INFO database is used.

- **Number of searches requesting help**: This number represents the total number of times a client process requested a search for "?" or "help". This gives you an idea of how many new users are requesting information about the databases served on this machine.
- **Avg. number of seed words per search**: This number is the sum of the number of words contained in all questions divided by the number of questions, or search requests. The word count also includes boolean operators and stopwords.
- **Number of searches using rel feedback**: This number is the total number of searches performed with relevance feedback.
- **Number of server warnings**: This is the number of times a warning occurred while processing a client's request.
- **Number of server errors**: This is the number of times an error occurred while processing a client's request.

In addition to these totals, the WAIS Reporter also compiles lists of four more items:

- The total number of search and retrieval requests for each database searched by a client process. This information gives you a quantitative idea of the load on each database provided by the WAIS server.
- The names of all client machines accessing the server's databases and the number of connections requested by each machine.
- The names of the client software and the number of connections requested by clients using this software.
- The error and warning messages of any problems reported by the server.

# How to Use the WAIS Reporter

The WAIS Reporter program is invoked with the following command sequence:

```
waisreporter log-file-name
```

where *log-file-name* is the pathname of the log file. For example, in your invocation of the WAIS server program, if you specified the **-e**

switch with the filename /wais/logs, the call to the WAIS Reporter might look like:

```
% waisreporter /wais/logs/server.log
```

Normally the WAIS Reporter sends the summary report to the standard output. If you want to save the report out to a file, you can redirect the output to the file of your choice.

```
waisreporter log-file-name > summary-file-name
```

where *summary-file-name* is the pathname of the output file.

# Options to the WAIS Reporter

If your WAIS server services multiple databases, and you would like to see a WAIS Usage Report of a specific database, you can use the **-d** switch to specify the database you want summarized:

```
waisreporter log-file-name -d database-name
```

where *log-file-name* is the pathname of the log file and *database-name* is the pathname of the database's index files. The *database-name* is the same name specified in the argument list to **waisindex**, **waislookup**, and **waissearch**. A sample usage follows:

```
waisreporter /wais/logs/server.log
             -d /wais/indexes/resumes
```

If the log file is stored as a compressed file, the - option is useful for piping the log information from the standard input. For example,

```
zcat compressed-file-name .Z | waisreporter -
```

This is particularly useful for analyzing archived log files.

# Sample WAIS Usage Report

An sample of a WAIS Usage Report is shown below.

```
WAIS Usage Report
generated on wais from /wais/logs/server.log-93-06-03
based on logs from Thu Jun  3 06:58:55 1993 to Thu Jun  3 18:42:26 1993

Total number of connections:                 175
Number of different machines connecting:       2 (87.50 connections each)
Total number of searches:                    113 (0.65 per connection)
Total connect time (seconds):              12741 (72.81 per connection)
```

```
Total search time (seconds):                391 (3.46 per search)
Searches returning zero hits:                 5 (4.42%)
Total number of documents retrieved:         84 (0.74 per search)

Total number of db's searched:                5
Number of searches with no DB name:           3 (2.65%)
Number of searches requesting help:           4 (3.54%)
Avg. number of seed words per search:      0.11
Number of searches using rel feedback:        0 (0.00%)
Number of server warnings:                    8
Number of server errors:                      0

Database                                                    Searches   Retrievals
-------------------------------------------------------------------------------
catalogs                                                          98           77
projects                                                           9            7
INFO                                                               3            0
resumes                                                            2            0
weather                                                            1            0
TOTAL                                                            113           84

Client Machine                                                        Connections
-------------------------------------------------------------------------------
saturn.wais.com                                                               173
jupitor.wais.com                                                                2
TOTAL                                                                         175

Client Software                                                       Connections
-------------------------------------------------------------------------------
                                                                                8
WAIS-INC-Mac-WAIS1.1-GN                                                         1
waissearch WAIS release 8 b2, from host: saturn.wais.com.                     165
waissearch WAIS release 8 b5.1, from host: saturn.wais.com                      9
TOTAL                                                                         175

Error and Warning Report
-------------------------------------------------------------------------------
none
```

# Automating Reporting

A simple shell script can be used to manage the log file and to periodically invoke WAIS Reporter. An example of such a script is included with your installation in the file /wais/current/util/run-waisreporter. This section describes the operation of this shell script, and describes how to use the UNIX **cron** facility to periodically invoke the script.

The script operates as follows. First the server.log file in the /wais/logs directory is moved to a new file whose name is based on today's date, (e.g. server.log-04-1-5.) Since the log file can become very large over time, segmenting it on a periodic basis helps to manage the size of the file. Next, an empty server.log file is created

as a replacement, and the permissions of this file are changed to ensure read and write accessibility. And finally, **waisreporter** is run on the logs contained in the old server.log file, and the resulting WAIS Usage Report is electronically mailed to the specified administrative personnel.

To set up a crontab entry that executes **run-waisreporter** on a periodic basis, first create a file, called *waisreporter-crontab-filename*, containing the following one-line entry[4]:

```
minute hour * * * run-waisreporter-filename -e logfile -m mailing-list
-b executable-directory
```

where *minute* is the minute of the hour (0-59), *hour* is the hour of the day (0-23), and *run-waisreporter-filename* is the full pathname of the shell script. This format specifies that the **cron** program should be run on a daily basis. The **run-waisreporter** script takes three arguments, the location of the log file, *logfile*, the mailing list to send the query report, *mail-address*, and the directory location of the **waisreporter** program.

For example, the following command specifies a crontab entry that will execute the script in /wais/current/util/run-waisreporter daily at 11:55 pm.

```
9 23 * * * /wais/current/util/run-waisreporter -e /wais/logs/server.log
-m wais-admin@wais.com -b /wais/current/bin
```

To set up your crontab entry, as superuser execute the following command:

```
% crontab -e user
```

where *user* is the name of a user account that will run the cron job. This will start an editor where you can enter the line above. To check that your crontab entry was set up properly, use the list option to **crontab**:

```
% crontab -l
10 23 * * * /wais/current/util/run-waisreporter -e /wais/logs/server.log
-m wais-admin@wais.com -b /wais/current/bin
```

To remove your crontab entry, use the **-r** switch to **crontab**.

---

[4]This example is based on a six field format. Check with your UNIX system administration manual to determine the exact format of your system's crontab.

# 8

# Reference Section

The following sections are the UNIX manual pages describing the WAIS programs They are available online in the /wais/current/man directory.

# waisindex

NAME

**waisindex** - index a stream of documents for fast search and retrieval

SYNOPSIS

**waisindex -d** *database-name*
**[ -mem** *megabytes* **]**
**[ -sw** *filename* **]**
**[ -a ]**
**[ -stem** *stemmer-name* **]**
**[ -nosrc ]**
**[ -nocat ]**
**[ -v ]**

DESCRIPTION

**waisindex** takes a stream of documents from stdin, typically generated by **waisparse,** and creates an index for fast search and retrieval of these documents. The final index requires approximately 1/3 to 1/2 of the memory space of the original data.

OPTIONS

**-d** *database-name* The **-d** switch specifies the pathname of the index files. For example, if /wais/indexes/foo is specified for *database-name*, then the index files will be located in the directory /wais/indexes/foo and named /wais/data/indexes/foo/index.*. If the directory, foo, does not exist, **waisindex** creates it. If the directory exists and already contains index files, the index files are overwritten. For indexing speed, *database-name* should be a directory on the local file system of the machine running **waisindex**. Indexing works over NFS, but it is significantly slower.

**-mem** *megabytes* The **-mem** switch specifies the number of megabytes of main memory the indexer can expect to use for indexing. The more main memory available to the indexer, the faster the indexing occurs. The default value is 10. For example, if the indexing machine has 64 or 256 megabytes of main memory, it may be reasonable to use a *megabytes* value of 50 or 200, respectively.

**-a** The **-a** switch tells the indexer to append to the current indexes in the database specified by *database-name*, instead of overwriting them. If the indexer encounters a document taken from a file that had not been previously indexed, it appends the document to the index. If a document is from a file that had been modified since the last indexing, the old document is deleted, and the newer version of the document is appended to the index. If a document if from a file that had not been modified since the last indexing, the indexer ignores it.

**-sw** *filename* The **-sw** switch specifies a file that contains a list of words to be used as stopwords, where each line of the file contains one stopword. Stopwords are ignored in the creation of the index. During natural language and relevance feedback search,

the stopwords have no effect. During boolean search, they are treated as if they have never occurred. The default stopword list is located in /wais/current/util/stopwords.txt, and may be modified as necessary.

**-stem** *stemmer-name* The **-stem** switch specifies the stemming algorithm to use on the database. Stemming is a technique used to automatically derive variations of a queried word. These variations are then used as part of the search. If a question contains the word "skate", for example, stemming is used to find documents that may also include "skates", "skated", and "skating". *stemmer-name* can be either **plural** or **porter**. Plural stemming searches for both the word and its corresponding plural. Porter stemming attempts to find the real base, or stem, of a word and derive any possible alternate variations. The default is no stemming.

**-nosrc** Inhibits the creation of a source file. If a source file already exists, the file is not overwritten, regardless of whether or not this option is given.

**-nocat** Inhibits the creation of a catalog. This option is useful for databases with a large number of documents, since the catalog contains 3 lines per document and may become large. A database is considered large when it holds approximately 1000 documents or more. A catalog is most useful for small databases, when a search returns no relevant documents, the catalog is returned.

**-v** Print the version of the **waisindex** program.

SEE ALSO

**waislookup**, **waisparse**, **waisreporter**, **waisserver**, **waissearch**, **waisworkstation**

# waislookup

NAME

**waislookup** - an interactive interpreter for searching and retrieving documents directly from a WAIS database

SYNOPSIS

**waislookup -d** *database-name*
**[ -v ]**

DESCRIPTION

**waislookup** accepts interactive search and retrieval requests for documents in a WAIS database. The search and retrieval is performed directly on the WAIS database without any intervention by a server process. **waislookup** is used primarily by the database administrator for checking for the correct construction of the WAIS database.

OPTIONS

**-d** *database-name* The **-d** switch is the full pathname for the database. It is the same name as specified in the **-d** switch to **waisindex**.

**-v** Print the version of the **waislookup** program.

USAGE

The **waislookup** interpreter provides five interactive commands: search, **r**etrieve, **f**eedback, **h**eadline, and **q**uit. By default, a question typed at the prompt followed by a carriage return is interpreted as a search command The question is either a natural language query, a boolean expression, or combination thereof. **waislookup** responds by returning a relevance-ranked list of documents from the database. Each document in the list is consecutively numbered with the document numbered 0 as the most relevant document.

After performing a search, the text of any of the listed documents can be retrieved by specifying the number of the document to be retrieved. Specifically, to retrieve a document numbered *n*, type an **r** character followed by one or more blank spaces, the integer value *n*, and a carriage return. Some documents may be available in multiple formats (the formats available are listed in each headline after its length). To retrieve a document in a specified format, specify the format *f* after the document number *n*. Thus, the command, "**r** *n f*" would retrieve document number *n* in format *f*. If there is no format specified, or the format specified is not valid for the document, the format defaults to the first format listed for the document.

Relevance feedback of a document numbered *n* is performed by typing an **f** character followed by one or more blank spaces, the integer value *n*, and a carriage return. The most important words of the specified document are then used to find additional documents that are "similar" to the original document. The default is to return all documents which matched the original question.

To limit the number of headlines returned by a search to *n*, at the command prompt type an **h** character followed by one or more blank spaces, the integer value *n*, and a carriage return. A value of -1 specifies that all documents should be returned.

To quit the **waislookup** interpreter, type a **q** character followed by a carriage return.

SEE ALSO

**waisindex**, **waisparse**, **waisreporter**, **waisserver**, **waissearch**, **waisworkstation**

# waisparse

NAME

**waisparse**- parse a database of files into a stream of documents for use by **waisindex**

SYNOPSIS

**waisparse** [ **-parse** *parse-format* ]
[ **-c**]
[ **-display** *display-format* ]
[ **-contents** |
[ **-nocontents** ]
[ **-assoc** *extension display-format* ]
[ **-wl** *length*]
[ **-nf** ]
[ **-ph** ]
[ **-v** ]
[ **-r** ]
*files-to-parse*

DESCRIPTION

**waisparse** takes a collection of documents in the form of files and directories, and separates the data into a stream of documents. For each document, the parser identifies a headline, field information, and the words. These are written to the stdout which is usually piped into **waisindex** to complete the building of a WAIS database. The original data is not modified.

OPTIONS

**-parse** *parse-format* The **-parse** switch tells the parser how to extract a document, headline, field information, and words from the files and directories in a data collection. The default parse format is **text**, where the complete file is the document, and the headline is the filename. To display a list of the parse formats supported in this release, run **waisparse** with no arguments. Each time a new **-parse** format is specified on the command line, any switches set up to that point are reset to their default value. This feature provides the capability of specifying different parse formats for different sets of data files.

**-c** The **-c** switch specified to **waisparse** prints a list of the parse formats contributed from the freeware community.

**-display** *display-format* The **-display** switch specifies the format of the documents in *files-to-parse*. **waisparse** associates a display format with each document encountered in *files-to-parse*. The display format is ultimately used by the WAIS client to determine how to display the document. When a parse format is not specified, the default display format is **TEXT**. Otherwise, the default is implied by the parse format. For example, if the parse format is **gif**, then the default display format is **GIF**.

**-contents** (**-nocontents**) Include (exclude) the contents of the file from the index. The filename and headline will always be included. The default is dependent on the parse format specified.

**-assoc** *extension display-format* The **-assoc** switch specifies an extension and display format of a secondary file that should be associated with the *files-to-parse*, where this secondary file is not parsed. This switch is especially useful for associating image files with text files together in a single document, where only the words in the text file are parsed. When a client performs a retrieval, the server returns both the text file and the image file. *extension* is a filename extension of the secondary file, and *display-format* is the display format of that file.

**-wl** *length* The **-wl** switch specifies the minimum number of characters that a word must contain to be parsed by **waisparse**. Words having less characters than the minimum word length are ignored by the parser and are not passed to the output stream. The default length value is 1.

**-nf** Do not parse field information. This option is useful if the selected parse format does supply field information and if it is necessary to reduce the storage requirements of the indexed files. The storage of field information increases index file size by roughly 25%.

**-ph** Print headlines for debugging new parsers.

**-v** Print the version of the **waisparse** program.

**-r** Recursively parse subdirectories.

*files-to-parse* This is a list of the pathnames of files to be parsed by **waisparse**. If the WAIS database is to be used from a machine other than the machine on which the parsing and indexing is performed, the filenames should be specified with a machine-independent pathname.

EXAMPLES

To parse and index a set of files, where the headlines are the filenames, and all the words are indexed and made into a database named manuals, execute the following:

```
% waisparse /product/manuals/*.doc.[ch] |
            waisindex -d /wais/indexes/manuals
```

To index a UNIX mail file containing many messages, where each message is a separately retrievable document, and the headline is the subject field of the mail message, execute the following:

```
% waisparse -parse mail-or-rmail /user/elmer/mail |
            waisindex -d /user/elmer/indexes/mail
```

To index Microsoft Word documents so they can be displayed on a PC or a Macintosh, the parse format is **text** and the display format is **MS-WORD**. Text format a the parse format where the filename is the headline, and the words of the file are put into the index. Although there may be special formatting instructions in the document, these extra words do not amount to much space in the

index, and do not effect search and retrieval. The display format of the documents is **MS-WORD** so that the client will know how to display it. The command is:

```
% waisparse -parse text -display MS-WORD
            /user/elmer/*.doc |
            waisindex -d /user/elmer/indexes/ms-doc
```

To index images where there are no words in the file, then only the filename can be used to provide terms for the index. The parse format can be specified in several ways. Either use the default parse format, **text**, and the **-nocontents** switch, or use the filename parse format:

```
% waisparse -nocontents -display GIF /user/elmer/*.gif |
            waisindex -d /user/elmer/indexes/weathermaps
or
% waisparse -parse filename -display GIF
            /user/elmer/*.gif |
            waisindex -d /user/elmer/indexes/weathermaps
```

Since the display format is **GIF**, a parse format of **gif** could have also been specified. In this case, the **gif** parse format implies a **GIF** display format. Therefore, another equivalent specification could be:

```
% waisparse -parse gif /user/elmer/*.gif |
            waisindex -d /user/elmer/indexes/weathermaps
```

An example of associating a set of text files with a set of gif and tif image files is:

```
% waisparse -assoc gif GIF -assoc tif TIFF
            /user/elmer/*.txt |
            waisindex -d /user/elmer/indexes/weathermaps
```

This says that a file named /usr/elmer/foo.txt, for example, is associated with a gif image file named /usr/elmer/foo.gif of display format GIF and a tiff image file named /usr/elmer/foo.tif with display format TIFF. Since the parse format is text, the headline is "foo".

Files with different parse types can be specified in the command line:

```
% waisparse -parse text -display MS-WORD /stooge/moe.doc
            -parse ps /stooge/joe.ps
            -parse filename -display MIME /stooge/curley
            | waisindex -d /stooge/indexes/info
```

This example illustrates the use of multiple parse formats within the same collection of data. It also shows that the **-parse** switch resets any subsequent switches to their default value. Specifying **-parse ps**, for example, reset the display format to the default, which is PostScript in this case.

SEE ALSO

**waisindex, waislookup, waisreporter, waissearch, waisserver, waisworkstation**

# waisreporter

NAME

**waisreporter** - generate a WAIS Usage Report

SYNOPSIS

**waisreporter** **[ -d** *database-name* **]**
**[ -sr ]**
**[ -ne ]**
**[ -nw ]**
**[ -pul ]**
**[ -v ]**
*log-file-name*

DESCRIPTION

**waisreporter** summarizes the contents of a log file generated by a WAIS server. The summary lists the number of searches and retrievals performed on each database, the total number of connections, the number of different machines connecting, the total connect time, the total search time, etc. It also lists any errors or warnings encountered by the WAIS server.

OPTIONS

**-d** *database-name* The **-d** switch specifies the name of the database that is to be summarized. If this argument is not specified, a summary of all databases recorded in the log file are generated.

**-sr** Print a detailed report on the searches performed.

**-ne** Suppresses reporting of any error messages from the log file.

**-nw** Suppresses reporting of any warning messages from the log file.

**-pul** Print any entries contained in the log file that **waisreporter** does not understand.

**-v** Print the version of the **waisreporter** program.

*log-file-name* This is the pathname of the log file that **waisreporter** is to summarize. If a '-' is specified, the log file is taken from the stdin, and the output goes to stdout.

# waissearch

NAME

**waissearch**- an interactive interpreter for sending search and retrieval requests to a WAIS server

SYNOPSIS

**waissearch** [ **-h** *host-machine* ]
[ **-p** *service-or-port* ]
[ **-d** *database-name* ]
[ **-m** *maximum-results* ]
[ **-v**]
*word...*

DESCRIPTION

**waissearch** is a client process that forwards search and retrieval requests to a WAIS server process running on the specified host machine. **waissearch** may execute on the same host machine as the WAIS server, or on a remote machine communicating via TCP. **waissearch** is used primarily by the database administrator for checking for the correct setup of the WAIS server, and for verifying the interprocess communication between the client and server processes. **waissearch** is a freeware software program (and is not available for Solaris).

OPTIONS

**-h** *host-machine* The **-h** switch specifies the name or IP address or hostname of the machine on which the WAIS server lives.

**-p** *service-or-port* The **-p** switch specifies the name of the WAIS service or port number used to communicate with the server machine.

**-d** *database-name* The **-d** switch specifies the name of the database the in which the server will search and retrieve information. If a full pathname is specified (for example, /wais/indexes/foo), the database at that location will be used. If only a name is specified (for example, foo), then the server looks for the database in the directory specified by the **-d** switch to **waisserver** or **waisworkstation**.

**-m** *maximum-results* The **-m** switch specifies the maximum number of documents that the server should return on a search request. The default is 40.

**-v** Print the version of the **waissearch** program.

*word...* This is a word or list of words constituting the user's question. It can consist of both natural language and boolean operators.

EXAMPLE

**waissearch** can be used as a client to contact the server that maintains the public directory-of-servers:

```
      waissearch -h quake.think.com -p 210
                 -d directory-of-servers help

SEE ALSO
      waisindex, waislookup, waisparse, waisreporter, waisserver,
      waisworkstation
```

# waisserver and waisworkstation

NAME

**waisserver**, **waisworkstation** - service WAIS search and retrieval requests

SYNOPSIS

**waisserver** [ **-p** *port* ]
[ **-d** *directory* ]
[ **-l** *loglevel* ]
[ **-e** *logfile* ]
[ **-u** *user* ]
[ **-v** ]

**waisworkstation** [ **-p** *port* ]
[ **-d** *directory* ]
[ **-l** *loglevel* ]
[ **-e** *logfile* ]
[ **-u** *user* ]
[ **-v** ]

DESCRIPTION

**waisserver** and **waisworkstation** are programs that service WAIS clients using the TCP protocol. For each new client that contacts the server, the server forks off a child server process to handle the client's requests. The child server remains active until the client closes the connection. If the server is started from the command line or from a shell script, the server is in standalone mode.

By convention, if the name of the process is **waisserver.d** or **waisworkstation.d**, then it is assumed to have been started from the **inetd** daemon. This mode of server operation is called daemon mode. See the examples below for how to set up the /etc/inetd.conf and /etc/services files.

OPTIONS

**-p** *port* The **-p** switch specifies the port number through which a client process should communicate with the server process. The default port number is 210. If the port number specified is less than 1024 (including 210), then the user must be root.

**-d** *directory* The **-d** switch specifies the base directory in which all the database directories for this server exist. If *directory* is not specified, and if the server is run in standalone mode, then *directory* defaults to the current working directory. If *directory* is not specified, and the server is run in daemon mode using the **inetd** daemon, *directory* defaults to /.

**-e** *filename* The **-e** switch specifies the name of the log file where the server records all transactions. If the server is run in standalone mode, the default is to send the log messages to stderr. If the server is run in daemon mode, the default is to send the log messages to /server.log.

**-u** *user* The **-u** switch specifies the user account which will run the server. You must be root to use the -u option. If no user is specified, the server runs as the user who started it.

**-v** Print the version of the **waisserver** or **waisworkstation** program.

EXAMPLES

The following are examples of **waisserver** usage. The usage for **waisworkstation** is identical. In standalone mode, the following command,

```
% waisserver -p 8000 -d /wais/indexes
             -e /wais/logs/server.log
```

runs using port 8000 on the databases in directory /wais/indexes logging messages in the file /wais/logs/server.log. For a WAIS client to communicate with this server, its communication port must also be set to 8000. Since the port number is greater than 1024, the **waisserver** command could be executed by any user.

To run in daemon mode, an example entry in the /etc/inetd.conf file is:

```
z3950 stream tcp nowait root /wais/current/waisserver
            waisserver.d -d /wais/indexes
            -e /wais/logs/server.log
```

Note this entry must be on one line in the /etc/inetd.conf file. To activate this new entry, send a HUP signal to the **inetd** process. In addition to modifying the /etc/inetd file, add the following line to the /etc/services file:

```
z3950 210/tcp  # Wide Area Information Server (WAIS)
```

This entry adds the new service, z3950, at port number 210 with the TCP protocol. At this point, if you are running NIS or NIS+, you may need to update your server.

SEE ALSO

**inetd**(8C), **inetd.conf**(5), **waisindex**, **waislookup**, **waisparse**, **waisreporter**, **waissearch**

# byte-split

NAME:

**byte-split** - splits a file into byte-sized chunks

SYNOPSIS:

**byte-split** *segment-size input-file*

DESCRIPTION:

Divides *input-file* into several chunks, each of *segment-size* or fewer bytes. **byte-split** reads input-file and writes it in segment-size pieces, as many as necessary, onto a set of output files in the current working directory. The name of the first output file is 'input-file' with '.0' appended, the next has '.1' appended and so on.

# ebcdic2ascii

NAME:

**ebcdic2ascii** - WAIS file format conversion utility

SYNOPSYS:

**ebcdic2ascii**

DESCRIPTION:

Converts ebcdic format to ASCII format. More specifically, it does the basic conversion of a-z, A-Z ,0-9, space, nl, and cr. Untranslated characters are replaced by '.'. Reads from stdin and writes to stdout.

EXAMPLE:

**% ebcdic2ascii < ebcdic-file > ascii-file**

# rmnl

NAME:

**rmnl** - WAIS file format conversion utility

SYNOPSIS:

**rmnl -dos**
**rmnl -single**

DESCRIPTION:

Converts DOS newlines to UNIX newlines, and converts double spaces to single space. Reads from stdin and writes to stdout.

OPTIONS:

**-dos** Convert from dos text format to ASCII text.
**-single** Convert from double to single space.

EXAMPLES:

**% rmnl -dos < dos-file > unix-file**
**% rmnl -single < double-space-file > single-space-file**

# tmpl

NAME:

**tmpl** - converts comma or tab delimited data into text a human-readable form

SYNOPSIS:

**tmpl -comma** *template-name*
**tmpl -tab** *template-name*

DESCRIPTION:

Reads a template file which describes how to format a comma or tab delimited line into a text representation suitable for use with WAIS. **tmpl** reads a line of input from stdin and breaks it into fields which are assigned consecutive numbers beginning with 1. It then prints the template file to stdout, substituting fields for variables defined in the template. Variables are indicated with printf-like syntax. A '%' sign followed by a format specifier and a field number makes up a variable. The format specifier for strings is 's'. Thus the line "%s3 City" would indicate that the third field should be printed followed by the string "City". Take for example the two comma-delimited input lines as follows:

```
Harry,Morris,30,not so hot
Dony,Morris,75,pretty good pretty good
```

And the template file:

```
-----------------------
# sample
Name: %s0 %s1
score: %s2
%s3
```

running the command:

```
% tmpl -comma t < t2
```

produces the following output suitable for parsing using the **dash** parse format:

```
-----------------------
Name: Harry Morris
score: 30
not so hot
-----------------------
Name: Dony Morris
score: 75
pretty good pretty good
```

BUGS:

The only format specifier supported is 's'.

# Appendix

# A

# WAIS Quick Start

So you are a UNIX wizard, and have decided to jump right in and install the WAIS software. Great! This section will show you what is needed and give a real life example. We'll keep the commentary to a minimum so you can bring your server up as soon as possible.

Note that in our examples, the current WAIS release is wais-1-0-10. The actual release which you are unpacking will have a different name, so whenever you are asked to type wais-1-0-10, replace it with the name of the version you are using.

### Installation

Let's get started. The first step is installing the WAIS software onto your UNIX machine. You'll need about5 megabytes of free space to install the software. For the examples in this section, we'll use /wais as our root directory. Once you've decided where to install the software, make the directory, and move into it.

```
% mkdir /wais
% cd /wais
```

If you've **ftp**'d the software over the Internet, you'll need to uncompress it and unpack it. The file is compressed if its filename ends in .Z, and it is an archive if it has a .tar extension. Here's the line that does work:

```
% mv wais-1-0-10.tar.Z /wais
% cd /wais
% zcat wais-1-0-10.tar.Z | tar -xvf -
```

If the software is on a tape, you'll have to copy the tape onto your hard disk. That means you will need to know which UNIX device describes the tape drive on your system. Your system administrator should be able to help with this. We'll use a typical one in this example; it is the first tape drive on the system and it's called /dev/rst0 on SunOS 4.x, and /dev/rdsk/c0t4d0s2 on SunOS 5.1

```
% cd /wais
```

```
% tar -xvf /dev/rst0
```

In either case, the computer should print something like this:

```
x wais-1-0-10/bin/waisindex, 344064 bytes, 672 tape blocks
x wais-1-0-10/bin/waisparse, 172032 bytes, 336 tape blocks
x wais-1-0-10/bin/waisserver, 745472 bytes, 1456 tape blocks
x wais-1-0-10/bin/waisreporter, 114688 bytes, 224 tape blocks
x wais-1-0-10/bin/waislookup, 458752 bytes, 896 tape blocks
x wais-1-0-10/bin/README, 106 bytes, 1 tape blocks
x wais-1-0-10/bin/waissearch, 507904 bytes, 992 tape blocks
x wais-1-0-10/util/rmnl, 73728 bytes, 144 tape blocks
x wais-1-0-10/util/byte-split, 24576 bytes, 48 tape blocks
x wais-1-0-10/util/ebcdic2ascii, 24576 bytes, 48 tape blocks
x wais-1-0-10/util/tmpl, 73728 bytes, 144 tape blocks
x wais-1-0-10/util/update-lcl-dir-of-servers, 1024 bytes, 2 tape blocks
x wais-1-0-10/util/run-waisreporter, 1226 bytes, 3 tape blocks
x wais-1-0-10/util/stopwords.txt, 2002 bytes, 4 tape blocks
x wais-1-0-10/util/README, 206 bytes, 1 tape blocks
x wais-1-0-10/doc/byte-split.txt, 689 bytes, 2 tape blocks
x wais-1-0-10/doc/rmnl.txt, 675 bytes, 2 tape blocks
x wais-1-0-10/doc/access-lists.txt, 1726 bytes, 4 tape blocks
x wais-1-0-10/doc/ethics.txt, 9338 bytes, 19 tape blocks
x wais-1-0-10/doc/waisworkstation.txt, 4817 bytes, 10 tape blocks
x wais-1-0-10/doc/waisindex.txt, 5659 bytes, 12 tape blocks
x wais-1-0-10/doc/waisparse.txt, 9185 bytes, 18 tape blocks
x wais-1-0-10/doc/waisserver.txt, 4817 bytes, 10 tape blocks
x wais-1-0-10/doc/ebcdic2ascii.txt, 588 bytes, 2 tape blocks
x wais-1-0-10/doc/tmpl.txt, 1814 bytes, 4 tape blocks
x wais-1-0-10/doc/waislookup.txt, 3416 bytes, 7 tape blocks
x wais-1-0-10/doc/waisreporter.txt, 1987 bytes, 4 tape blocks
x wais-1-0-10/doc/waissearch.txt, 3414 bytes, 7 tape blocks
x wais-1-0-10/man/byte-split.1, 531 bytes, 2 tape blocks
x wais-1-0-10/man/ebcdic2ascii.1, 450 bytes, 1 tape blocks
x wais-1-0-10/man/rmnl.1, 557 bytes, 2 tape blocks
x wais-1-0-10/man/tmpl.1, 1486 bytes, 3 tape blocks
x wais-1-0-10/man/waisindex.1, 4103 bytes, 9 tape blocks
x wais-1-0-10/man/waislookup.1, 2525 bytes, 5 tape blocks
x wais-1-0-10/man/waisparse.1, 7345 bytes, 15 tape blocks
x wais-1-0-10/man/waisreporter.1, 1489 bytes, 3 tape blocks
x wais-1-0-10/man/waissearch.1, 2771 bytes, 6 tape blocks
x wais-1-0-10/man/waisserver.1, 3723 bytes, 8 tape blocks
x wais-1-0-10/sample/README, 3335 bytes, 7 tape blocks
x wais-1-0-10/sample/data/resumes/res3.txt, 4102 bytes, 9 tape blocks
x wais-1-0-10/sample/data/resumes/README, 5243 bytes, 11 tape blocks
x wais-1-0-10/sample/data/resumes/res1.txt, 3070 bytes, 6 tape blocks
x wais-1-0-10/sample/data/resumes/res2.txt, 4634 bytes, 10 tape blocks
x wais-1-0-10/sample/data/mail/rmail, 5760 bytes, 12 tape blocks
x wais-1-0-10/sample/data/mail/README, 4394 bytes, 9 tape blocks
x wais-1-0-10/sample/data/jargon/README, 4292 bytes, 9 tape blocks
x wais-1-0-10/sample/data/jargon/convert.c, 1874 bytes, 4 tape blocks
x wais-1-0-10/sample/data/jargon/jargon.231.text, 558266 bytes, 1091 tape blocks
x wais-1-0-10/sample/data/weather/ACUS1.DOC, 4961 bytes, 10 tape blocks
x wais-1-0-10/sample/data/weather/README, 2866 bytes, 6 tape blocks
x wais-1-0-10/sample/data/weather/WXKEY.DOC, 31 bytes, 1 tape blocks
x wais-1-0-10/sample/data/weather/WXMAPARS.DOC, 36 bytes, 1 tape blocks
x wais-1-0-10/sample/data/weather/ACUS1.GIF, 13581 bytes, 27 tape blocks
x wais-1-0-10/sample/data/weather/SELSLOG.GIF, 13870 bytes, 28 tape blocks
x wais-1-0-10/sample/data/weather/WXKEY.GIF, 18865 bytes, 37 tape blocks
x wais-1-0-10/sample/data/weather/WXMAPARS.GIF, 35661 bytes, 70 tape blocks
x wais-1-0-10/sample/data/weather/SELSLOG.DOC, 5216 bytes, 11 tape blocks
x wais-1-0-10/sample/Makefile, 1327 bytes, 3 tape blocks
x wais-1-0-10/INSTALL, 11796 bytes, 24 tape blocks
x wais-1-0-10/README, 1916 bytes, 4 tape blocks
```

When you set up your WAIS server and WAIS databases, you won't want to encode the version number in each configuration file, since that would mean reconfiguring each file when a new release is installed. Instead we recommend making a symbolic link which maps the name of the release directory into a standard name used by the configuration files. In our example, we'll use "current" as the standard name.

```
% ln -s wais-1-0-10 current
```

When a new release is installed (e.g. wais-1-0-11), you can switch over to the new software by removing the old link, and making a new one.

To install the WAIS programs and UNIX manual pages, add the following to your .cshrc file:

```
setenv PATH /wais/current/bin:$PATH
setenv MANPATH /wais/current/man:$MANPATH
```

and type the following at the command line

```
% source ~/.cshrc
```

The installation applies to C shell (csh) users only. If you use another shell (e.g. borne shell, korn shell, etc.), you will need to modify installation to suit your shell.

Now let's take a look at the software just installed. The **ls** command does a directory listing.

```
% ls current
bin   doc   man   sample   util
```

The bin directory contains the WAIS executables., the doc directory contains documentation, the man directory has the UNIX-formatted manual pages, the sample directory has sample databases to experiment with, and the util directory has useful conversion programs and shell scripts.

This completes the installation of the WAIS software.

## Building an Example WAIS Database

Now you are ready to build a WAIS database. A WAIS database is made up of a collection of data, and a WAIS Index. The installation includes some sample collections of data from which you can try

building and searching databases. Let's move into the sample directory, and see what it contains.

```
% cd current/sample
% ls
Makefile   README   data
% ls data
jargon   mail   data   weather
```

The /wais/current/sample/data directory contains several small sample data collections for you to practice with. Let's now create a WAIS database using the data in the resumes directory. First, let's take a look at our data collection.

```
% ls data/resumes
README   res1.txt   res2.txt   res3.txt
```

The resumes directory contains three resume files, where the first line of each file contains the name. You may want to use your editor or the UNIX **more** command to view the contents of the files.

```
% more data/resumes/res1.txt
```

Now that you have a sample collection of data, the next step is to build the WAIS index. If this is the first time you are attempting this, you will need to create an index directory:

```
% cd /wais/current/sample
% mkdir indexes
```

A WAIS index is created from your original collection of data using the **waisparse** and **waisindex** programs. **waisparse** takes a collection of documents in the form of files and directories, and separates the data into a stream of documents. For each document, the parser identifies a headline and the content-bearing words. **waisindex** takes a stream of documents from **waisparse** and creates an index for fast search and retrieval of these documents.

The command line for building an index of the resumes data is a single line (no carriage returns) like:

```
% waisparse -parse first-line
            /wais/current/sample/data/resumes/*.txt |
            waisindex -d /wais/current/sample/indexes/resumes
```

The method of parsing is specified by the word after the -parse switch. This tells the parser that each file contains one document and that the headline is the first line in each file. The vertical bar "|" feeds the output of **waisparse** into **waisindex**. It's called a pipe in

UNIX jargon. **waisindex** reads the data from **waisparse** and builds a database, whose name is specified by the argument following **-d** (/wais/current/sample/indexes/resumes in this example). The output you receive should look like the following:

```
23 0: Jun  3 15:36:53 1993: 7: parsing /wais/current/sample/data/resumes/res1
24 0: Jun  3 15:36:53 1993: 100: indexing /wais/current/sample/data/resumes/res1
25 1: Jun  3 15:36:53 1993: 7: parsing /wais/current/sample/data/resumes/res2
26 1: Jun  3 15:36:54 1993: 100: indexing /wais/current/sample/data/resumes/res2
27 2: Jun  3 15:36:54 1993: 7: parsing /wais/current/sample/data/resumes/res3
28 2: Jun  3 15:36:54 1993: 100: indexing /wais/current/sample/data/resumes/res3
29 3: Jun  3 15:36:55 1993: 100: flushing 1312 words (929 different words) to disk
30 4: Jun  3 15:36:56 1993: 100: generating headers
31 5: Jun  3 15:36:58 1993: 100: wrote source description
/wais/current/sample/indexes/resumes/resumes.src, please examine it before use
32 6: Jun  3 15:36:58 1993: 100: building catalog
```

This completes the building of your WAIS database.

You've now built your first WAIS database. Let's check the results, and then run a query against it. The **waislookup** program provides an easy way to check that a database is built properly. It contains all the same search capabilities as a WAIS server, but doesn't use the network to serve the data. Instead, the user is talking directly to the database, which must be on a locally mounted disk. **waislookup** takes the name of the database following the **-d** switch. An sample interactive search session follows:

```
% waislookup -d /wais/current/sample/indexes/resumes
interactive search - q to quit, h # to set number of headlines (-1 =
all)
                        r # to retrieve, f # to feedback
> UNIX AND programming
33 score 1598 len  4102 type TEXT  John William Emmerson
> r 0
34 1: Jun  3 15:38:30 1993: 5; retrieving docID: "0 -4102
/wais/wais/current/sample/data/resumes/res3" bytes 0 4102 from database "" type "TEXT"
                      John William Emmerson
                       35 Camry Avenue
...
> q
%
```

The **waislookup** program puts you into an interactive session in which you can ask questions of the information stored in the database. In this example, the first interactive command typed at the prompt is a question made up of the words "UNIX" and "programming" separated by the boolean operator "AND". This tells the **waislookup** program to look for documents containing both the words "UNIX" and "programming". **waislookup** returns the headline of one document (numbered 0) that satisfied this criteria. The second

interactive command is a retrieval request for the document numbered 0. **waislookup** returns and prints the contents of the document. To return to the UNIX prompt, use the **q** command.

Try running the sample **waislookup** session yourself to verify that your resumes database has been set up properly. Once you have completed this step, you are now ready to set up a server process.

## Setting up a WAIS Server

The WAIS server handles search and retrieval requests from clients. The server programs are called **waisserver** and **waisworkstation**. The remainder of this section demonstrates the usage of the **waisserver** program; the usage of the **waisworkstation** program is identical. The command line looks like:

```
% waisserver -p 8000 -d /wais/current/sample/indexes &
36 0: Jan 5 16:09:43 1904: 100: running server WAIS Inc
```

The **-p** switch gives the TCP port number that the server should use to communicate with client processes. The **-d** switch tells the server to find its databases in /wais/current/sample/indexes. The **&** at the end of the command line tells the shell to run the command in the background. This allows you to type other commands at the prompt while the server process is running.

Once the server process has been started, try running a search from a client. You may want to first try running the **waissearch** client on the same machine as the server, and then try it from a remote machine. If possible, run it from a different shell process, since the server prints out the record of each client/server transaction. A sample **waissearch** session on the resumes database follows:

```
% waissearch -p 8000 -d resumes UNIX AND programming
 Search Response:
  NumberOfRecordsReturned: 1
   37 Score: 1000, lines:  -1 ' John William Emmerson'

View document number [type 0 or q to quit]: 1
Headline:  John William Emmerson
                           John William Emmerson
                             38 Camry Avenue
...
View document number [type 0 or q to quit]: q
Search for new words [type q to quit]: q
```

Note that the **waissearch** program is provided by the freeware community and currently only runs under Sun 4.1.x and NeXTstep 3.x. Other client programs are also available through freeware and are included with your software installation, or via anonymous **ftp** from wais.com.

When the server responds to requests from a client process, the server prints out information to the shell. Your server output from processing the above **waissearch** session should look like:

```
39 1: Jun  3 17:49:07 1993: 1: accepted connection from: wais [192.216.46.98]
40 2: Jun  3 17:49:07 1993: 100: child PID = 5271
41 0: Jun  3 17:49:08 1993: 100: init message: waissearch WAIS release 8 b5.1, from
host: wais.wais.com
42 1: Jun  3 17:49:08 1993: 3: search database: "resumes" seed words: "UNIX AND
programming"
43 2: Jun  3 17:49:08 1993: 3: Checking access for client 192.216.46.98 to database
/wais/current/sample/indexes/resumes/index.acc
44 3: Jun  3 17:49:08 1993: 4: returned 1 results
45 4: Jun  3 17:49:08 1993: 4: results: 0 -4102 /wais/current/sample/data/resumes/res3
46 5: Jun  3 17:49:13 1993: 3: Checking access for client 192.216.46.98 to database
/wais/current/sample/indexes/resumes/index.acc
47 6: Jun  3 17:49:13 1993: 5: retrieving docID: "0 -4102
/wais/current/sample/data/resumes/res3" bytes 0 4102 from database "resumes" type
"TEXT"
48 7: Jun  3 17:49:17 1993: 2: done handling client
```

To kill the server process, first determine the process number (PID), and use the UNIX **kill** command.

```
% ps -auxww | grep waisserver
margaret     5270   0.0  2.2 2.16M  360K p4 S     0:00 waisserver -p...
margaret     5274   0.0  1.3 1.52M  208K p4 S     0:00 grep waisserver
% kill 5270
49  Terminated waisserver -p 8000 -d /wais/current/sample/indexes
```

# B

# Recommended Reading

The books listed here are ones we have found helpful. Most of them involve UNIX system administration.

Frisch, AEleen: *Essential System Administration*, O'Reilly & Associates, Inc., Sebastopol, CA 1991. Covers UNIX on a variety of platforms.

Nemeth, Evi, Garth Snyder, and Scott Seebass: *UNIX System Administration Handbook*, Prentice Hall Software Series, Englewood Clifs, NJ 1989. Good general UNIX system administration background, a bit outdated.

Stallman, Richard: *GNU Emacs Manual*, 6th ed., Freeware Software Foundation, Cambridge, MA, 1987. Emacs is our preferred UNIX editor. It's freely available, and much more powerful than vi, but also harder to learn.

Winsor, Janice: *Solaris System Administrator's Guide*, Ziff-Davis Press, Emeryville, CA 1993. If you are administering a Solaris machine, this book is absolutely essential.

# C

# Default Stopword List

The following words make up the default stopword list used by **waisparse**. They are included with your release in the file /wais/current/util/stopwords.txt. Stopwords are words which occur so frequently that they are not useful for distinguishing one document from another. Since they aren't useful for searching, they are not indexed.

a
about
above
according
across
actually
adj
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
among
amongst
an
and
another
any
anyhow
anyone
anything
anywhere
are
aren
around
as
at
be
became
because
become
becomes
becoming
been
before
beforehand
begin
beginning
behind
being
below
beside
besides
between
beyond
billion
both
but
by
can
cannot
caption
co
could
couldn
did
didn
do
does
doesn
don
down
during
each
eg
eight
eighty
either
else
elsewhere
end
ending
enough
etc
even
ever
every
everyone
everything
everywhere
except
few
fifty
first
five
for
former
formerly
forty
found
four
from
further
had
has

hasn
have
haven
he
hence
her
here
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
how
however
hundred
ie
if
in
inc.
indeed
instead
into
is
isn
it
its
itself
last
later
latter
latterly
least
less
let
like
likely
ll
ltd
made
make
makes
many
maybe
me
meantime
meanwhile
might
million
miss
more
moreover
most
mostly
mr
mrs
much
must
my
myself
namely
neither
never
nevertheless
next
nine
ninety
no
nobody
none
nonetheless
noone
nor
not
nothing
now
nowhere
of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
overall
own
per
perhaps
rather
re
recent
recently
same
seem
seemed
seeming
seems
seven
seventy
several
she
should
shouldn
since
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
stop
such
taking
ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thirty
this
those
though
thousand
three
through
throughout
thru
thus
to

together
too
toward
towards
trillion
twenty
two
under
unless
unlike
unlikely
until
up
upon
us
used
using
ve
very
via
was
wasn
we
we
well
were
weren
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever
whole
whom
whomever
whose
why
will
with
within
without
won
would
wouldn
yes
yet
you
your
yours
yourself
yourselves

# D

# Glossary of WAIS Terms

**.acc**

File extension for access files. An access file contains the IP addresses of all machines that are allowed to search the database. It is created by the database administrator to control access to a database.

**.cat**

File extension for catalog files. See also catalog.

**.dct**

File extension for dictionary files. A dictionary file contains the dictionary of all the words used in a database.

**.doc**

File extension for document table files. A document table file contains a record of each document in the database.

**.fn**

File extension for filename table files. A filename table file lists the filenames of the original data files.

**.hl**

File extension for headline table files. A headline table file contains the headlines of all the documents in the database.

**.inv**

File extension for inverted files. An inverted file lists of all the words in the database, and for each word all the documents which contain that word.

**.qst**

File extension used by question files. See also question structure.

**.src**

File extension used by source description files which describe the database and the server. They are the means by which a client contacts the server and searches the database. Source description files are distributed to clients by the directory-of-servers. They are ASCII files, and can be edited by hand. See also source structure.

**boolean operator**

A boolean operator is a mathematical operator based on set theory. Boolean operators provide a very powerful mechanism for specifying exact relationships between words in a WAIS question. Each boolean

operator denotes a very specific relationship between sets of words. For example, the question "dog AND cat" contains the boolean operator "AND", and specifies that only documents having both the word "dog" and the word "cat" will be retrieved. In terms of set theory, this question is the intersection between the sets defined by dog and the sets defined by cat. Other Boolean operators include OR, ADJ and NOT.

## catalog

Catalog (.cat) files contain a human readable list of headlines and document id's for some or all of the documents in the database. This list may be returned to a user whose search has gone poorly, as an aid to help them understand the contents of the database.

## client

In a client/server architecture, the client is the program which requests services. With WAIS, clients are user interface programs which request services from remote or local services, using the WAIS protocol.

## daemon mode

This is the name for the mode the server is in when it is run by the **inetd** network daemon. See also standalone mode.

## database

is a service on a server that answers questions based on some defined expertise. It is described by a .src file that can usually be retrieved by asking "help" of the server now.

## Directory of Servers

A database that serves .src files of WAIS databases. The public Directory of Servers on the Internet is located at quake.think.com on port 210.

## display format

The display format specifies how a client will display a retrieved document. By default, the display format is determined by the parse format. See also parse format.

## document identifier

(aka doc-id) A unique string that identifies each document in a database.

## field

For data collections whose documents are structured in a semi-regular format, the regular portions of the documents can be tagged by the WAIS parser as fields. For example, in an electronic mail message, the 'to' and 'from' words are fields.

## fielded search

Performing a restricted search based on the value of field or set of fields is called fielded search.

## gwais

**gwais** is a WAIS freeware client program for GNU-Emacs developed by Jonathan Goldman at Thinking Machines (jonathan@think.com).

## inetd daemon

The **inetd** daemon is the Internet network daemon. This is a UNIX program that manages the network services according to the configuration specified by the /etc/inetd.conf file. The **inetd** daemon runs in the background and manages network requests by spawning off other daemons, or programs, to service these requests.

## parse format

A parse format determines how **waisparse** breaks a data file into its component documents, and decides which words to use as a headline. See also display format.

## phrase matching

If the user's question contains a natural language expression made up of more than one word, called a phrase, a phrase-matching technique is employed. Using this technique, a higher weight is assigned to a document containing a phrase that identically matches the phrase in the user's question.

## plural stemming

Plural stemming is a stemming algorithm that attempts to derive the root form of a word if given a plural.

## port

A port number specifies a service on a UNIX machine. For example, WAIS usually runs on port 210, and Telnet usually runs on port 23. Ports below 1,000 are called "well-known" ports, and you must be superuser to run a program on them. The mapping between the service name and port number is determined in /etc/services file.

## porter stemming

Porter stemming is a stemming algorithm that attempts to find the real base, or stem, of a word and derive any possible alternate variations.

## proximity relationship

A proximity relationship designates that if the words in a question are located close together in a document, they are given a higher weight than those found further apart. The idea behind a proximity relationship is that words found in close proximity to each other in a document more likely contain the same content as that specified in the user's question.

## query

See question.

## query report

A query report is a document created by the server that describes how a client's question is parsed by the server.

## question structure

A question structure is a file format used by some client programs to store the state of a user's inquiry so that it can be used again. It includes

what databases to ask, the user's question, and any relevant documents. It may also store the last list of results.

## question

A question is an expression containing a combination of natural language words and boolean operators.

## relevance feedback

Relevance feedback is the ability to select a document or a portion of a document and find a set of documents "similar" to the selection. In essence, relevance feedback adds more words to the original question. It uses the most important words and phrases in the relevant document in addition to the original question. The most important words and phrases are determined by the same weighting algorithms as the words in the original question. The weight of the relevant document terms is less than the original question terms.

## relevance ranking

Relevance ranking is the scoring of documents based on their relevance to a user's question, where the most relevant document has the highest score, or rank. A document receives a higher score if the words in the question are in the headline, or if the words appear many times, or if the phrases occur exactly as in the question. A document's score is derived from using techniques such as word weighting, term weighting, phrase matching, proximity relationships, and word density.

## right truncation

A user can specify right truncation in a question by ending a word on the right with the wildcard character, '*'. This tells the WAIS server to search on words matching the base characters before the '*'. An example of using right truncation is a question such as "geo*", which may retrieve documents containing the words: geographer, geography, geologist, geometry, geometrical, etc.

## server

In a client/server architecture, the server is the program which provides the services. With WAIS, a server is a machine that supports databases and answers questions.

## Solaris

(aka. SunOS 5.x) Sun's version of AT&T's System V Release 4 UNIX operating system.

## source structure

A source structure is a file format used by WAIS client programs to describe a particular database on a particular server. Typical contents are the machine name, ip address, port of the server, and the name of the database.

## standalone mode

A WAIS server is run in standalone mode when the **waisserver** program is invoked directly by the user or shell script. See also daemon mode.

### stemming

Stemming is a technique used to automatically derive the root of a queried word. The root is then used to search against the roots of the words contained in a database. If a question contains the word "skate", for example, stemming is used to find documents that may also include "skated" and "skating".

### stopword

A stopword is a word that occurs so frequently that it is not useful for distinguishing one document from another. Since it is not useful for searching, it is not indexed.

### superuser

On machines running the UNIX operating system, the superuser is a privileged user who has access to all files and all services provided by the machine. This status is usually reserved for system administrators.

### swais

**swais** is a freeware WAIS client program for ASCII terminals developed by John Curran.

### TCP/IP

TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol. This is the low level protocol which is used on the Internet, and on many LANs. It provides reliable data communication.

### term weight

Each word used in a database is assigned a numerical value, called the term weight, based on the frequency of occurrence of that word over all documents in the database. Words that occur frequently throughout the database are not weighted as highly as the words that appear less frequently. Very common words are either ignored or diminished in the scoring. For example, since the term, "animal", may occur frequently in many of the documents in a database, its term weighting is small compared to a term such as "hippopotamus", which may occur in only a small number of documents.

### WAIS

WAIS is an acronym for Wide Area Information Servers and a trademark of WAIS Inc.

### WAIS database

A WAIS database consists of a set of documents, a set of index files, and a source file. See also database.

### WAIS protocol

The WAIS protocol is used to connect WAIS clients and servers. It is based on the Z39.50 protocol. Because a standard protocol is used clients and servers can be built on a wide variety of computer, architectures communicating over local and wide-area networks.

**waisindex**
This program takes the documents specified by waisparse and builds a WAIS searchable index.

**waislookup**
This is a very simple WAIS client, which provides a dumb terminal interface directly to WAIS indexes. It does not go through the protocol or server, rather it does the search internally. It is mainly used for troubleshooting WAIS indexes.

**waisparse**
This program reads a data file and breaks it into its component documents, decides which words to index, and which words to use as the headline. The output of waisparse is fed to waisindex.

**waisreporter**
this program summarizes the log files generated by waisserver. It can be used to monitor database usage.

**waissearch**
This is a very simple WAIS client, which provides a dumb terminal interface to any WAIS server on the network. It is mainly used for troubleshooting of WAIS servers.

**waisserver**
This program waits for a connection from a client, and handles the connection by searching and/or retrieving documents from a WAIS database.

**WAIStation**
**WAIStation** is a WAIS client program for the Macintosh. WAIStation is a trademark of Thinking Machines Corporation. See also *WAIS for Macintosh User's Manual.*

**word density**
The ratio of the number of times a queried word appears in a document to the size of the document is called the word density. It is a measure of how important a queried word is to the overall content of the document. A higher word density results in a higher relevance ranking.

**word weight**
If a word in a document is found to match a word in the user's question, the word is assigned a word weight, and this weight additively contributes to the overall score of the document. The exact weight that a word receives depends on where in the document the word was found. A word is weighted highest if it appears in the headline, less if the word appears in all capital letters or if the first letter of the word is capitalized, and finally, a word has the least weight if it appears only in the text.

**xwais**
**xwais** is a freeware WAIS client program for X-Windows developed by Jonathan Goldman at Thinking Machines (jonathan@think.com).

## Z39.50

Z39.50 is the National Information Standards Organization (NISO) protocol for information search and retrieval.

# Index